







Rapport de stage

Développement du curateur d'Aniseed

Réalisé par

Marlène GUILLEMETTE

Sous la direction de

Cyril MARTIN

Pour l'obtention du DUT informatique

Année universitaire 2012-2013

Remerciements

Avant toute chose j'aimerais remercier Patrick Lemaire et Cyril Martin de m'avoir acceptée pour ce stage et de m'avoir accordé leur aide, leur gentillesse et leur disponibilité au cours de ces trois mois de stage.

Je remercie également toute l'équipe Lemaire pour l'accueil, la bonne humeur et l'ambiance agréable tout au long du stage. Merci à l'équipe de biologistes de m'avoir donné l'occasion de voir des embryons d'ascidies ainsi que des étoiles de mer. Merci également à Mme Gelsomino pour ses conseils sur la conception de ce rapport.

Pour finir, je voudrais remercier mon tuteur M. Fabien Michel, pour son aide et ses conseils, ainsi que l'ensemble du corps enseignant de l'Institut Universitaire et Technologique de Montpellier pour les enseignements que j'ai reçus depuis deux ans. J'aimerais remercier tout particulièrement mes professeurs de programmation qui m'ont appris des notions de POO, qui m'ont été très précieuses durant ce stage.

Sommaire

Table des figures

Glossaire

<u>Notions informatiques :</u>

IDE (Environnement de développement logiciel) : Programme regroupant un ensemble d'outils pour le développement de logiciels. Il regroupe généralement un éditeur de texte, un compilateur, des outils automatiques de fabrication et un débogueur.

Ontologie : Une ontologie est un système contenant des termes, une définition de ces termes, ainsi qu'une spécification des relations existant entre les termes, formant ainsi un vocabulaire contrôlé. Une ontologie et généralement représentée par un graphe, les nœuds étant les termes et les arcs les relations entre ces termes.

Système de gestion de base de données relationnelle et objet : Ensemble de logiciels qui servent à manipuler des bases de données.

Généricité : Indépendance vis-à-vis du type

Framework : Ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns, l'ensemble formant ou promouvant un squelette de programme.

DAO : (Objet d'accès aux données) est un patron de conception utilisé dans les architectures logicielles objet.

Smarty : Smarty est un moteur de template pour le langage PHP.

ORM : Technique de programmation faisant le lien entre le monde de la base de données et le monde de la programmation objet. Elle permet de transformer une table en un objet facilement manipulable via ses attributs.

<u>Notions biologiques :</u>

Clone : L'ADN situé dans le noyau est isolé et coupé avec un enzyme de restriction. Tous les fragments d'ADN sont insérés individuellement dans un vecteur pour produire plusieurs millions de molécules recombinantes. Ces molécules sont propagées chez des bactéries pour produire les clones. Lors d'une expérience de type in situ hybridization, le biologiste va utiliser une sonde qui reconnaît spécifiquement le gène recherché. Cette sonde se fixe sur le gène s'il est présent dans la cellule et le signale par sa fluorescence, sa radioactivité, etc.

In situ : Comparaison d'un embryon sauvage pour un stade donné et du même embryon ayant subi une perturbation.

Biomatériel : Un biomatériel est un embryon. Il est dit « perturbé » s'il a été perturbé par une expérience permettant la manipulation de la fonction d'un gène, ou de l'organisation des cellules embryonnaires.. Sinon c'est un embryon à l'état sauvage n'ayant subi aucune modification. Une fois le stade d'analyse du biomatériel sauvage défini, ce stade est le même pour les biomatériels « perturbés » correspondants.

Profils d'expression : Un profil d'expression est la description de l'état d'activité d'un gène (son expression) dans chaque cellule d'un embryon à un stade donné.

Présentation de l'entreprise

Le CNRS

Le Centre National de Recherche Scientifique est un organisme public français de recherche créé en 1939. Ce centre est classé comme établissement public à caractère scientifique et technologique. Il est présidé par Alain Fuchs, assisté de deux directeurs généraux délégués, Joël Bertrand à la science et Xavier Inglebert aux ressources. 54 laboratoires sont associés au CNRS de la délégation du Languedoc-Roussillon, dont l'essentiel est implanté à Montpellier.

Le CNRS exerce son activité dans tous les champs de la connaissance en s'appuyant sur plus de 1100 unités de recherche et de service, dont le CRBM.

Le CRBM

Le Centre de Recherche de Biochimie Macromoléculaire, créé en 1968, est à l'origine une unité propre du CNRS qui est devenue, début 2007, une unité mixte de Recherche associée aux Universités Montpellier 1 et 2. La mission première du CRBM est de développer une recherche fondamentale de pointe et de réputation internationale dans les domaines de la biologie moléculaire.

L'équipe Lemaire

L'équipe « Contrôle transcriptionnel de la morphogenèse des chordés », dirigée par Patrick Lemaire, se compose de biologistes et d'informaticiens. Son objectif général est de comprendre comment la séquence linéaire du génome dirige la formation dynamique d'un animal multicellulaire complexe, en utilisant comme système modèle les embryons d'un proche parent des vertébrés, l'ascidie Ciona Intestinalis. Les ascidies ont été choisies en raison de leur simplicité anatomique et génomique. De plus, les ascidies montrent une excellente conservation des lignages cellulaires embryonnaires, des destinées cellulaires et des formes de cellules. Les ascidies sont donc très adaptées pour déchiffrer le programme de développement d'un chordé, et pour comprendre comment la stabilité du développement est codée dans l'ADN génomique.

Pour atteindre ses objectifs, l'équipe combine des approches embryologiques, moléculaires et bioinformatiques. Pour intégrer les données hétérogènes de ce projet (séquences nucléotidiques diverses, territoires anatomiques, données d'expression génique…) dans des conditions sauvages ou mutantes, l'équipe a conçu et maintient toujours le projet Aniseed (Ascidian Network for In Situ Expression and Embryological Data). Il est composé d'une base de données et de plusieurs applications web. La version 3 est disponible à l'adresse suivante : http://www.aniseed.cnrs.fr/

La version 4 est actuellement toujours en développement. Mon stage concerne la refonte de l'interface de curation de cette nouvelle version. Il comprend plusieurs aspects comme l'adaptation à la nouvelle base de données, le développement suivant un framework, l'amélioration de l'ergonomie de l'interface…

Introduction

L'équipe de Patrick Lemaire travaille sur quatre espèces d'ascidies : Ciona intestinalis, Ciona savignyi, Halocynthia roretzi et Phallusia mammillata. Afin d'enregistrer leurs données génomiques, l'équipe d'informaticiens à été chargée de créer la base de données Aniseed. Cette base de données dispose de plusieurs outils de fouille de données. Parmi eux, le curateur permet aux biologistes de consulter et d'ajouter de nouvelles informations dans la base de données.

La version 3 d'Aniseed dispose d'une interface de curation qui, bien qu'à peu près fonctionnelle, a été codée sans utiliser de framework (ni de design pattern ni de POO), et avec une base de données ne correspondant plus à celle utilisée dans la quatrième version.

C'est pourquoi l'objectif de mon stage est de recréer l'interface de curation tout en l'adaptant à la quatrième version, donc en la faisant correspondre à l'actuelle base de données et au framework Jelix. La nouvelle version profitera également de formulaires plus faciles à utiliser grâce à l'utilisation de JavaScript, d'Ajax et d'autres outils visant à améliorer l'interactivité.

Dans un premier temps, une présentation du contexte et des contraintes sera faite à l'aide du cahier des charges. Dans cette partie, vous trouverez un résumé des besoins fonctionnels et des spécifications techniques. Ensuite j'aborderai les notions techniques du stage et j'expliciterai les choix de conception. Après cela, je présenterai le manuel d'utilisation expliquant comment utiliser le site et permettant de voir le résultat concret qui a été obtenu. Enfin, je résumerai tous les points liés à l'activité en explicitant le mode de développement adopté et le déroulement du stage.

1. Cahier des charges

1.1. Analyse du contexte

1.1.1. Le sujet

« Amélioration de l'application web Aniseed : développement du curateur. »

Il s'agit en fait de développement web utilisant le framework Jelix, une base de données PostgreSQL et des notions en JavaScript et en Jquery.

1.1.2. Le projet Niseed

Le projet Niseed (Network for In Situ Expression and Embryological Data) est un système informatique permettant de représenter l'ensemble du développement d'un animal à travers différents types de données biologiques. Niseed met à disposition de l'utilisateur une gamme étendue d'outils de fouille qui permettent l'extraction et la création de nouvelles données.

L'une des forces de ce projet est sa généricité* : l'idée est de pouvoir étendre ce système à plusieurs groupes d'organismes, c'est-à-dire pouvoir intégrer des données indépendamment des organismes étudiés. Aniseed est la version consacrée au groupe des ascidies qui sont des invertébrés marins (Ascidian Niseed). Un historique des différentes versions d'Aniseed est visible dans l'annexe 1. C'est la plus récente version d'Aniseed encore en développement aujourd'hui qui va donner naissance au système générique.

Le projet se compose de différents modules (cf. figure 1) :

- Deux bases de données PostgreSQL (une pour Niseed, une pour Gbrowse)
- GBrowse : Outil open source développé en Perl et disponible sur le site de la communauté GMOD. Combinant base de données et page Web interactive, il permet la manipulation et l'affichage d'annotations de génomes.
- Le module Niseed du manager : Application web qui permet à un administrateur d'installer une nouvelle base. De plus, elle fournit une gamme d'outils qui permettent de gérer l'ensemble des utilisateurs, de maintenir à jour les données de la base, et de créer de nouvelles informations.
- Le module Niseed public, aussi appelé navigateur développemental : Application web qui met à disposition de l'utilisateur différents outils de fouille de données. Il est organisé autour de divers domaines : données d'expression, données anatomiques, données moléculaires…
- Le module Niseed du curateur : Application web proposant des outils d'insertion et d'édition de données spécifiques à la fonction du bio curateur. Un bio curateur est une personne primordiale en biologie car elle est censée accréditer ou discréditer les informations contenues dans la base de données. De son travail dépend la crédibilité des données publiées. Le curateur peut contrôler, modifier, annoter et supprimer certaines données présentes dans la base à travers cette application.

▫ 3D virtual embryo : Application JAVA permettant de visualiser les données contenues dans la base aniseed, dans le contexte d'embryons virtuels reconstruits en 3D.

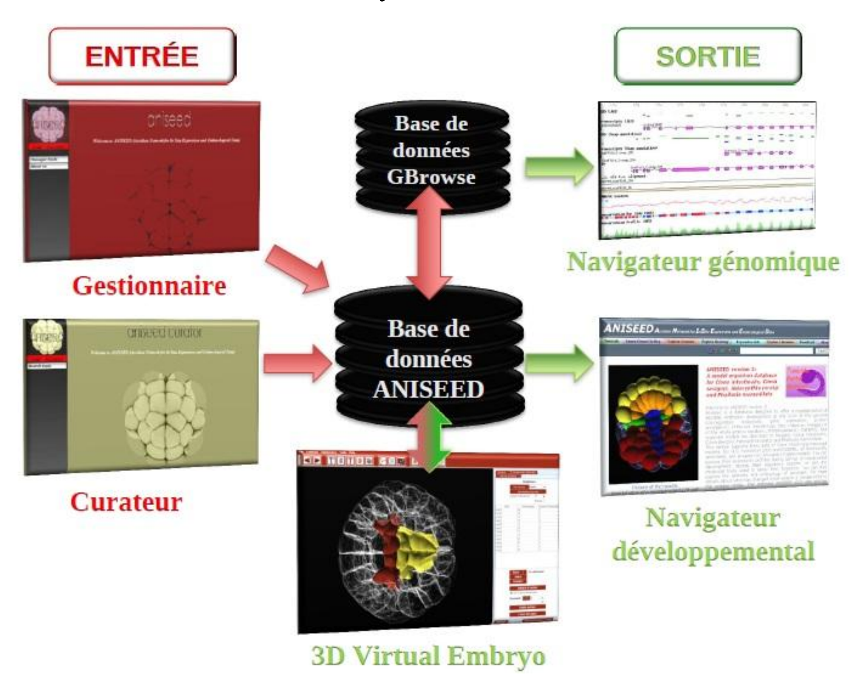


Figure 1 - Architecture du projet Aniseed

1.1.2. Qu'est ce que la curation ?

La curation est utilisée par des sites qui souhaitent améliorer la lisibilité de leur contenu. En effet, la curation permet de faire gagner du temps aux internautes lors de leur recherche : le tri et la sélection de l'information ont déjà été opérés et les résultats affichés correspondent à la requête, la curation permet également d'organiser les grandes quantités d'information et de faire émerger des contenus parfois peu connus. Elle repose sur cinq étapes fondamentales :

→ Première étape : Recensement des types de données à insérer
→ Deuxième étape : Définition des formats de représentation de ces types de données.
→ Troisième étape : Identification des jeux de données
→ Quatrième étape : Saisie des données et vérification de leur qualité et intégrité
→ Cinquième étape : Partage au public des données

Les objectifs de la curation sur le site d'Aniseed sont multiples :

- La curation permet d'extraire les données provenant de la littérature, ainsi que de connecter les informations de différentes sources de manière cohérente et compréhensible.
- Elle permet l'informatisation des données présentes dans les publications en extrayant l'information pertinente.
- Elle permet également de définir et gérer un vocabulaire contrôlé.
- Grâce à la curation, on peut corriger les erreurs présentes dans la représentation de données.
- Enfin, la curation va permettre de remplir la base de données Aniseed et de ce fait aider les utilisateurs à se servir et à trouver les données dont ils ont besoin.

Dans le cas d'Aniseed, les annotateurs et curateurs vont d'abord lire et recenser des informations pertinentes pour le site. Pour cela, ils pourront soit récupérer les informations directement dans l'équipe où ils travaillent, soit trouver ces informations dans des publications scientifiques, recensées sur le site pubmed[1].

Ensuite, ils effectueront une structuration de ce contenu. Par exemple, pour enregistrer un nouvel outil moléculaire récemment apparu, ils définiront l'espèce impliquée, le gène, le type de régulation… La structuration est cruciale pour l'informatisation de ces données.

La dernière étape concerne le partage de ce contenu. Le curateur pourra rendre disponible et accessible les nouvelles données aux internautes.

Le curateur d'Aniseed jouera également un tout autre rôle: Il devra valider (ou discréditer) les informations saisies par les annotateurs qui sont d'autres utilisateurs de l'interface de curation.

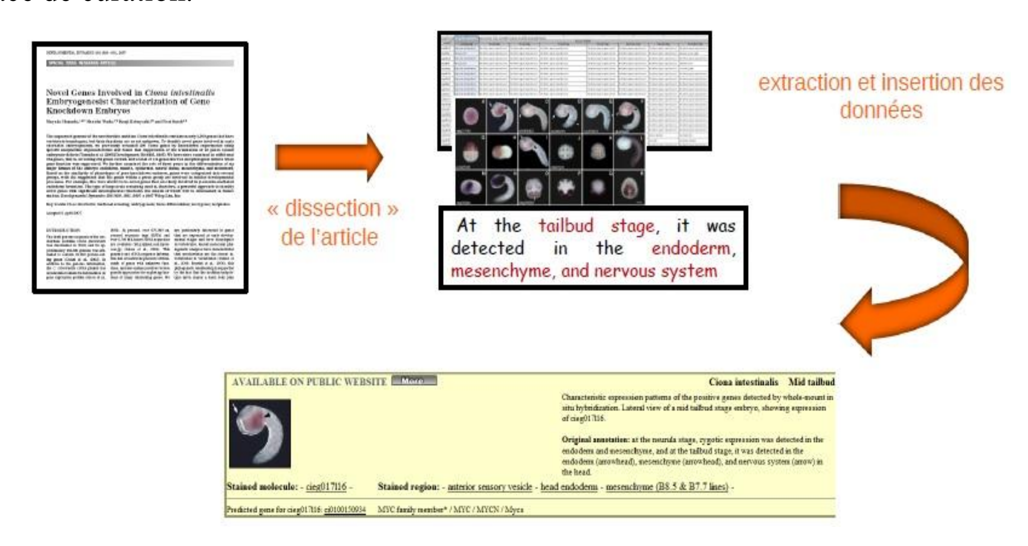


Figure 2 – Le rôle du curateur d'Aniseed

1.1.3. Description de l'existant

La version 3 d'Aniseed permet de répondre aux besoins des biologistes mais pèche par la qualité de son architecture logicielle, ce qui la rend difficilement modifiable et portable. Elle est composée d'environ 800 scripts mis à la racine du projet, sans suivre de design pattern et sans utiliser de POO. Cette version est également basée sur une base de données peu optimisée : certaines tables n'ont pas de clé primaire, il y a des redondances dans les données…

Actuellement, la version 4 d'Aniseed est en plein développement. Cette nouvelle version est destinée à faciliter la lecture, la compréhension et la maintenance de l'application en utilisant des concepts informatiques comme la programmation orientée objet, le design pattern MVC et le framework* Jelix. De plus, une migration de base de données est en cours. En effet, les anciennes versions utilisaient une base de données qui évoluait au gré des modifications et des besoins pressentis par les auteurs. La nouvelle base de données PostgreSQL suit le schéma Chado[2] qui a été mis en place par la communauté GMOD[3] (Generic Model Organism

[1] http://www.ncbi.nlm.nih.gov/pubmed
[2] http://gmod.org/wiki/Chado_-_Getting_Started

Database). Ce modèle est spécifique aux bases de données biologiques, ce qui le rend particulièrement adéquat au système Aniseed.

1.2. Analyse des besoins fonctionnels

1.2.1. Quels sont les utilisateurs de l'interface de bio curation?

Tous les utilisateurs de l'interface de curation sont des biologistes. Il y a deux catégories d'utilisateurs : les annotateurs et les curateurs.

Les annotateurs sont des utilisateurs disposant de moins de droits que les curateurs : ils peuvent interagir avec l'interface de curation pour ajouter des données et les modifier. Leurs données sont automatiquement enregistrées comme privées. C'est-à-dire que seules les personnes de leur laboratoire y auront accès et que la qualité de leurs données ne sera pas contrôlée. S'ils veulent que les données qu'ils ont enregistrées deviennent publiques, ils peuvent les soumettre à curation. Un bio curateur se chargera alors de définir si la qualité des données soumises est suffisante pour les insérer dans la base de données publique. Les données validées seront ensuite disponibles pour tous les internautes sur l'interface publique. Par contre, si une donnée n'est pas validée par le bio curateur, elle ne sera pas visible sur l'interface publique.

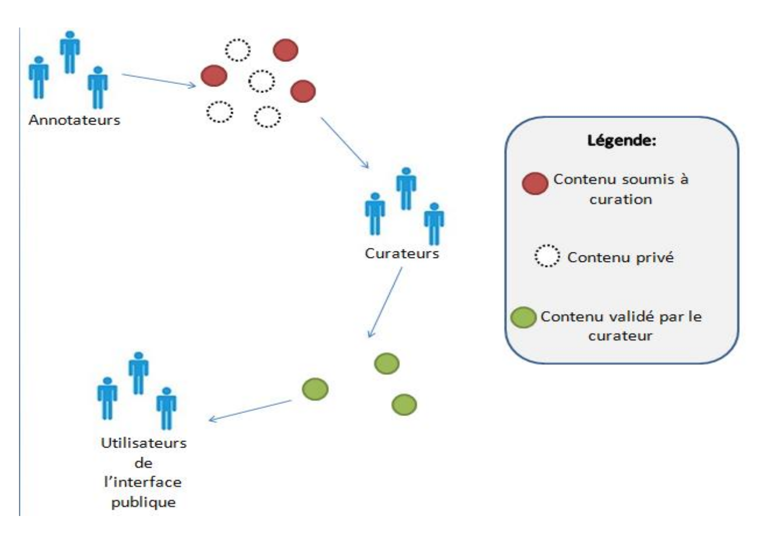


Figure 3 - Le deuxième rôle du curateur

Les curateurs, en plus de bénéficier des droits de base, pourront annoter les données, en supprimer… Ils sont également chargés de créditer ou discréditer les informations entrées par les autres utilisateurs en contrôlant leur validité. Pour l'instant, la seule curatrice d'Aniseed est Delphine Dauga.

Pour le développement de l'interface de curation à faire pendant le stage, dû à une contrainte de temps, il n'y aura pas besoin de gérer les droits, on considérera qu'il n'y a que des curateurs.

1.2.2. De quoi ont-ils besoin ?

Les utilisateurs de l'interface de curation ont des données de plusieurs types à rentrer dans la base. Ils auront par exemple des images à attacher à certains objets biologiques, des liens à faire entre un gène et une publication, ou des simples chaînes de caractères à enregistrer.

La gestion des liens pourra être facilitée : pour augmenter la rapidité du travail de l'utilisateur, on pourra faire des tris de données. Par exemple : un outil moléculaire est lié à un

[3] http://gmod.org/wiki/Main_Page

gène ainsi qu'à des publications. L'outil moléculaire ne pourra être associé qu'à des publications en lien avec le gène choisi… Quand l'utilisateur va créer un outil moléculaire, après qu'il a saisi son gène, on lui proposera uniquement les publications qui sont liées à ce gène. L'information minimale requise pour l'interprétation biologique des données insérées dépend du type de données considéré. Pour l'insertion d'un « outil moléculaire » (un type de réactif biologique qui permet de modifier la fonction d'un gène), on aura besoin des données présentes dans la figure 4 ci-dessous.

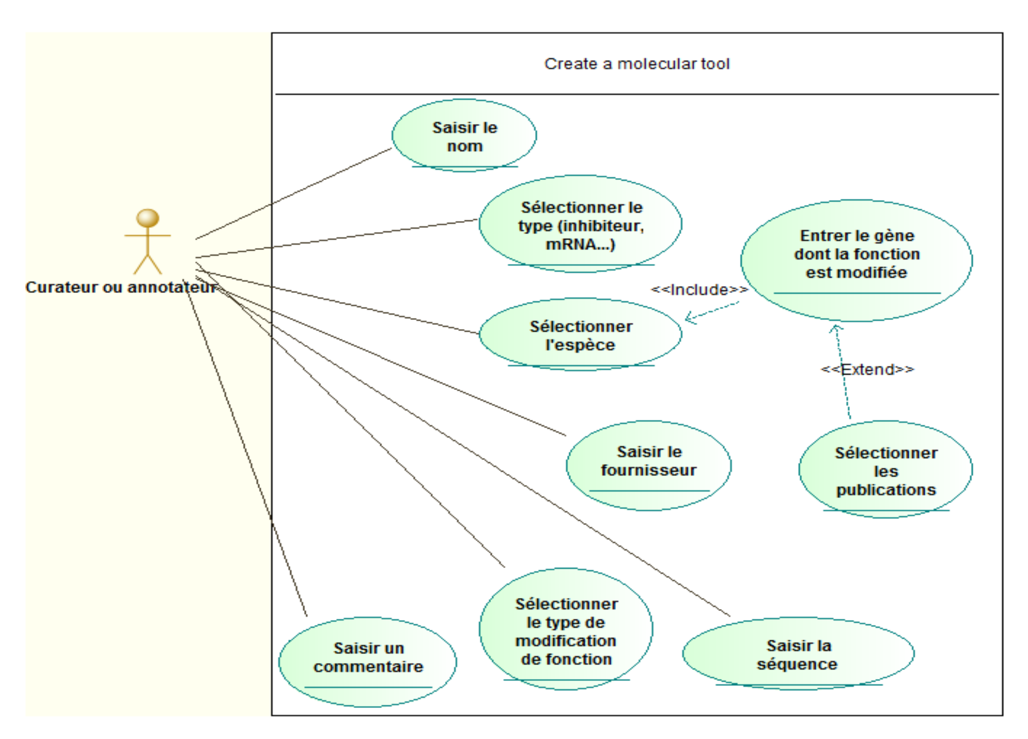


Figure 4 - UseCase de création d'un outil moléculaire

Sur la page de création d'une expérience de description du profil d'expression d'un gène (expérience dite « in situ »), l'utilisateur va devoir rentrer de multiples informations. En fait, la création de l'in situ se fera en plusieurs étapes. Au total l'utilisateur doit pouvoir rentrer toutes les informations citées dans la figure 6.

Ces informations concerneront plusieurs objets. Il y aura une expérience, liée à des biomatériels (nécessairement un seul biomatériel de type sauvage, et zéro, un ou plusieurs biomatériels de type perturbé) pouvant chacun être associé au profil d'expression d'un ou plusieurs gènes. Ces objets et leurs relations sont visibles dans la figure 5 ci-dessous.

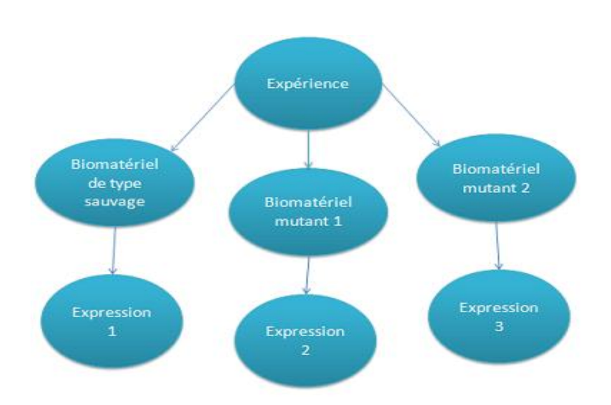


Figure 5 - Plusieurs objets forment une expérience de type in situ

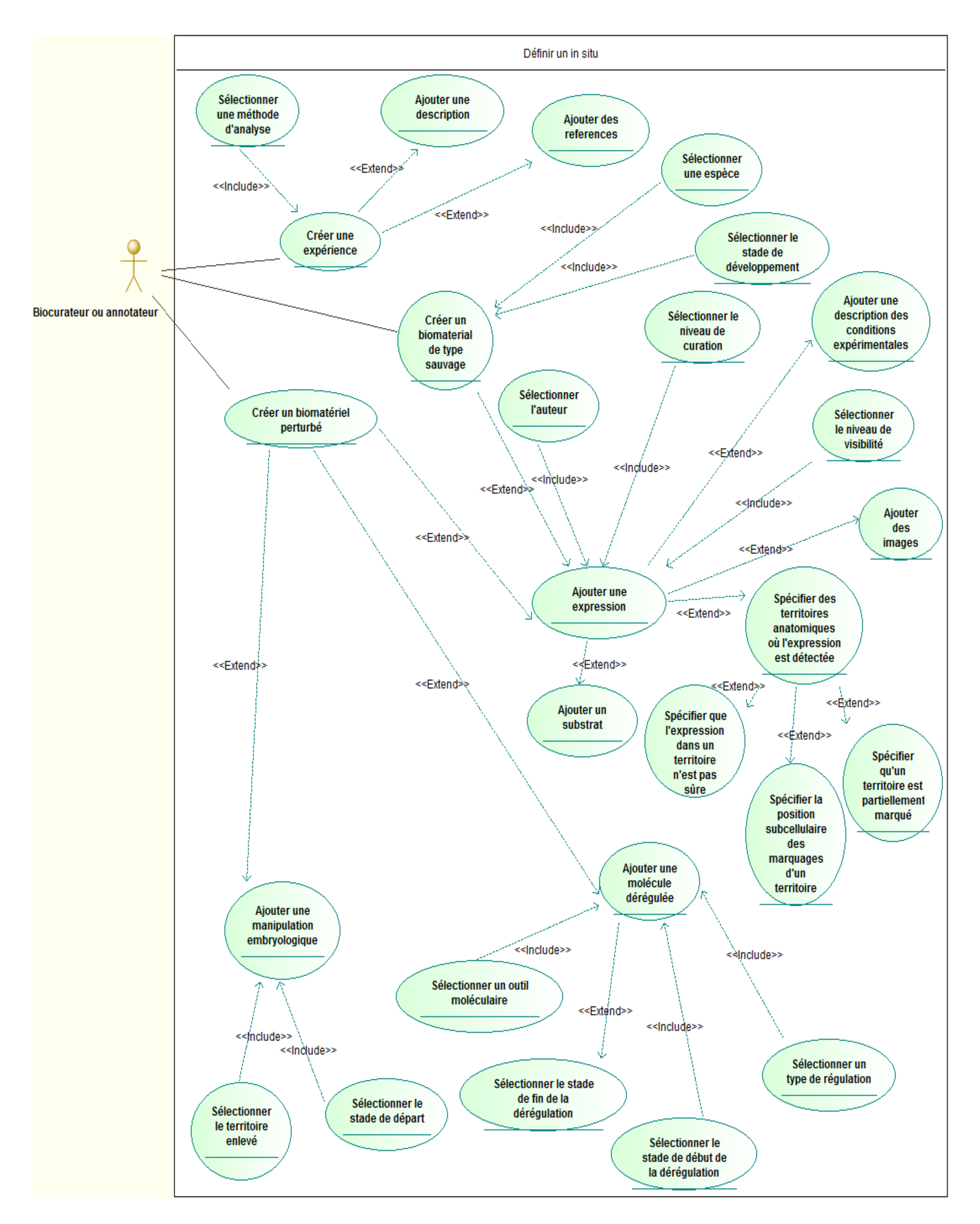


Figure 6- Use Case de création d'un in situ

1.3. Analyse des besoins non fonctionnels

1.3.1. Environnement de travail

- Système d'exploitation : Linux
- Environnement de développement intégré : NetBeans IDE* 7.2
- Outils de développement :
  - Serveur web : Apache 2.2
  - Langages : HTML, PHP 5, JavaScript, XML
  - SGBD : PostgreSQL 9.0

1.3.2. Aspects ergonomiques

Les utilisateurs de l'interface de curation sont des biologistes qui, pour la plupart, n'ont suivi aucune formation en informatique. Il faut donc créer un outil facile d'utilisation, où la navigation se fait de façon intuitive. De plus, les curateurs sont rares et doivent de ce fait pouvoir effectuer leur travail rapidement. L'interface doit faciliter le travail de ces personnes tout en restant simple à prendre en main.

1.3.3. Organisation du travail

Il faudra rendre compte régulièrement de l'avancement de son travail sur Redmine, qui est un gestionnaire de projets utilisé par toute l'équipe Lemaire. Le compte rendu devra être rédigé en anglais. Un cahier m'a été donné dans lequel j'ai décidé de tenir un carnet de bord.

Il y aura aussi des présentations orales du projet à faire lors de réunions d'équipe (deux pendant la totalité du stage). Les réunions d'équipe se déroulent en anglais et ont lieu tous les mardis pendant deux heures. Ces présentations devront bien entendu être faites en anglais et être accompagnées d'un diaporama.

2. Rapport technique

2.1. Contexte technique

2.1.1. La base de données

Une base de données contenant des données biologique est une base de données originale de par sa conception et l'architecture de ses tables. En effet, les données biologiques sont atypiques et nécessitent de ce fait un traitement un peu particulier car elles sont parfois incomplètes, erronées ou périmées. De plus, ces données vont évoluer et s'échanger très rapidement. Pour pallier ces problèmes, les concepteurs de Niseed proposent une annotation manuelle ou semi-automatique des données. Cette facilité de mise à jour de la base garantit une meilleure qualité des informations stockées.

De plus, Aniseed insiste sur la traçabilité des données. Pour chaque information saisie, on va retenir son auteur, les publications où elle est citée, son annotateur, la date de saisie, et le bio curateur l'ayant validée. Cette traçabilité permet de mieux savoir l'origine des données.

Un autre point important dans la base de données d'Aniseed concerne les hiérarchies pour la représentation des relations de filiation. Dans mon équipe d'accueil, et dans la biologie du développement en général, on s'intéresse aux embryons et leur description se fait par des hiérarchies composées de termes issus d'un vocabulaire contrôlé. Une ontologie* (exemple dans la figure ci-dessous où les liens orientés sont de type « fait partie de ») regroupe l'ensemble du vocabulaire contrôlé et de la hiérarchie.

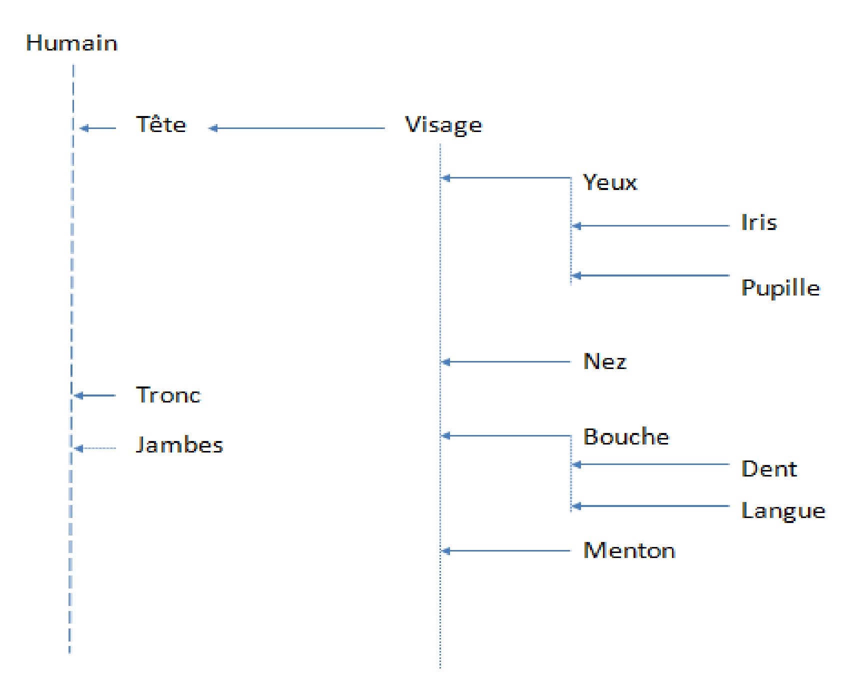


Figure 7 – Un exemple de dictionnaire anatomique hiérarchisé.

Aniseed repose sur une base de données de modèle Chado. Chado permet la standardisation des bases de données contenant des informations biologiques. En effet, il est basé sur l'utilisation extensive d'ontologies.

Le choix des ontologies par les concepteurs de Chado a été motivé par les raisons suivantes :

- Intégrité des données : les vocabulaires contrôlés empêchent l'utilisation de valeurs différentes pour désigner une même chose et permettent d'éviter d'éventuelles fautes d'orthographe ou de syntaxe.
- Portabilité et standardisation des données : des ontologies internationales ont été créées dans le but de permettre un partage plus efficace des résultats entre chercheurs (il existe plusieurs ontologies très connues en biologie : *Gene Ontology* qui réunit toutes les caractérisations des gènes, *Sequence Ontology* qui regroupe toutes les caractérisations des types de séquences rencontrées dans un génome.)
- La maintenabilité : elle résulte des points abordés ci-dessus.

C'est l'un des schémas relationnels les plus sophistiqués existant actuellement en biologie moléculaire. Il a été utilisé pour le site d'Aniseed car il est capable de contenir la plupart des classes rencontrées en biologie. De plus, Chado et Jelix sont complémentaires du fait qu'ils sont tout deux basés sur un système de « modules ». Chado est un schéma modulable ce qui en fait un outil flexible et extensible.

Les modules les plus caractéristiques sont les suivants :

- Controlled vocabulary (cv) : Vocabulaires contrôlés et ontologies
- Organism : Données taxinomiques
- Sequence : Objets basés sur des séquences
- Publication : Publications scientifiques et références
- Genetic : Données génétiques et les génotypes
- Mage : Données de microarray
- Companalysis : Informations issues d'analyses informatiques

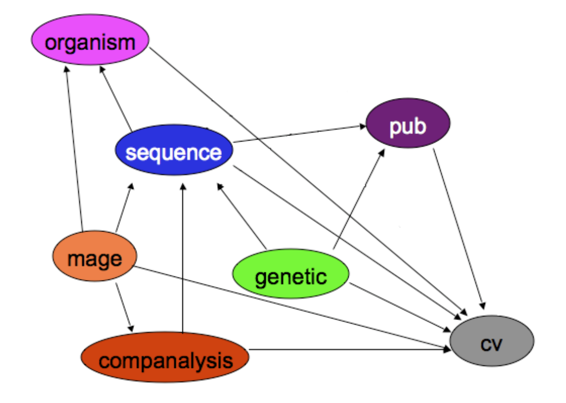


Figure 8 - Modules caractéristiques de Chado

Nous allons prendre l'exemple du module cv et voir comment sont architecturées ces tables afin de mieux illustrer la stratégie générale utilisée dans Niseed.

Les différentes tables du module cv (cf. figure 9) permettent de stocker les termes définis (cvterm) ainsi que des propriétés s'y rattachant (cvtermprop), les synonymes des termes (cvtermsynonym), les relations qui les unissent (cvterm_relationship), les bases de données d'où proviennent les termes (db), les références externes (dbxref), les propriétés qui s'y rattachent (dbxrefprop), et enfin la distance séparant des cvterm liés entre eux (cvtermpath).

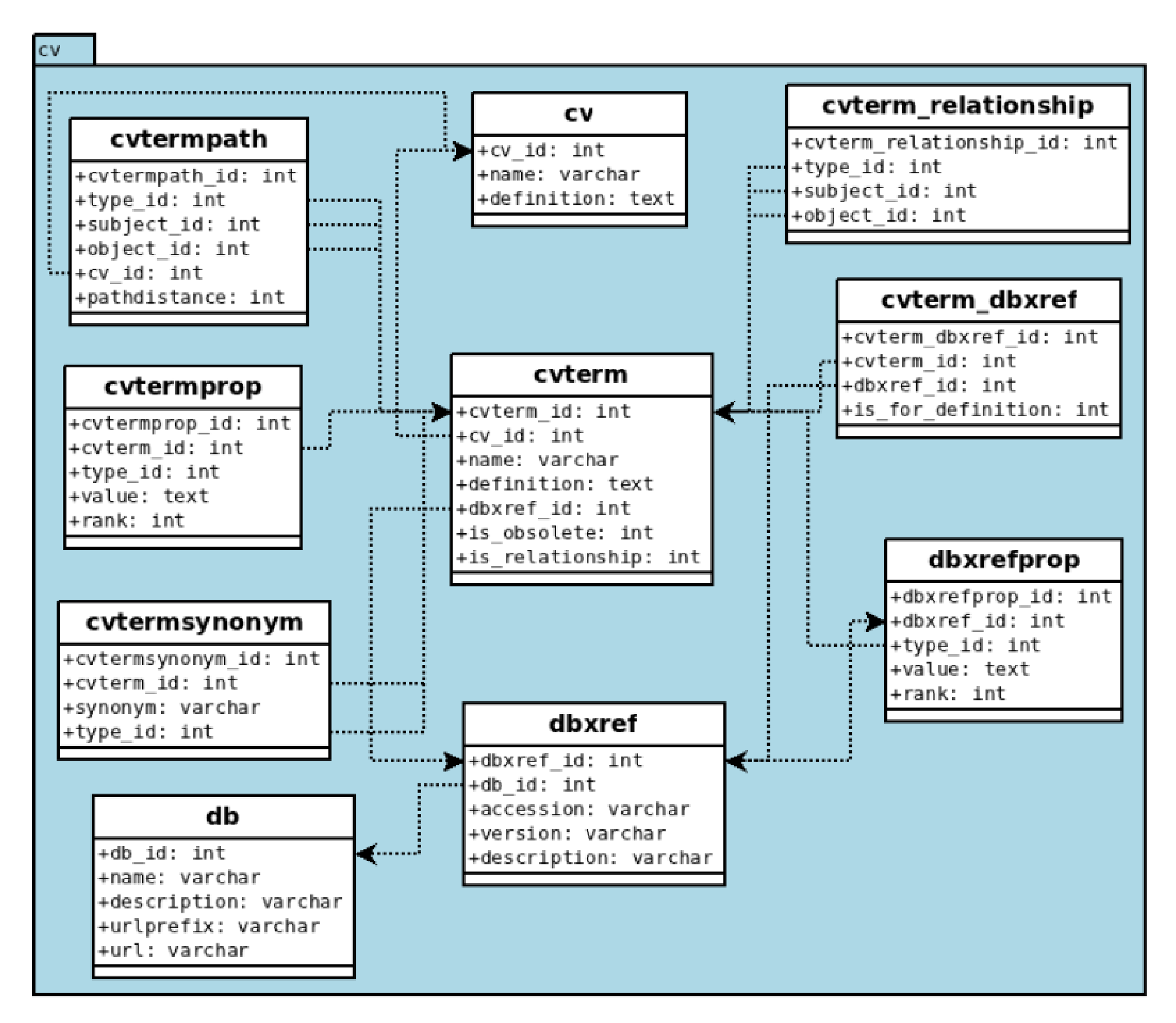


Figure 9 - Diagramme de navigation entre les tables pour le module cv

Le schéma Chado est implémenté avec le SGBD PostgreSQL 9.0

PostgreSQL est un système de gestion de base de données relationnelle et objet*. Ce système est fondé sur une communauté mondiale de développeurs et d'entreprises. Il propose plusieurs outils aidant au développement : psql, une interface en ligne de commande permettant la saisie de requêtes SQL, pgAdmin est un outil d'administration graphique et enfin PhpPgAdmin qui est une interface web d'administration.

PostgreSQL offre une particularité (partagée avec Oracle) particulièrement intéressante : le partitionnement de la base en schémas. Un schéma est un espace de noms : il contient des objets nommés dont les noms peuvent être identiques à ceux d'objets d'autres schémas. Les objets nommés sont accessibles en préfixant leur nom de celui du schéma. Cette division est particulièrement intéressante pour Aniseed car la base contient des informations multi-espèces, de ce fait on peut faire un partage logique des données en créant un schéma par espèce.

Enfin, bien qu'elle suive le modèle Chado, la base de données d'Aniseed comporte des modules additionnels. En effet, Chado fut initialement conçu pour représenter des données génomiques basées sur la séquence de l'ADN. L'originalité de Niseed est d'avoir étendu la

logique de Chado à des données non basées sur des séquences d'ADN (par exemple les données anatomiques, les molécules chimiques…)

Aniseed contient les données suivantes :

→ Données anatomiques : l'anatomie, le destin, le lignage, la morphologie…
→ Littérature
→ Données génomiques : Les clones, les gènes et l'annotation, les séquences cis-régulatrices, les outils moléculaires…
→ Données d'expression des gènes

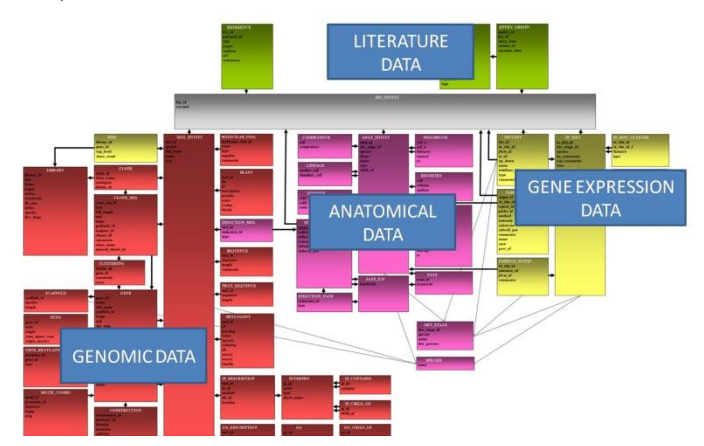


Figure 10 - Aperçu de la base de données

Ces données sont toutes répertoriées dans les modules suivants : Aniseed, anatomy, companalysis, contact, curator, cv, experiment, expression, library, mage, manager, nd, news, organim, phenotype, pub, sequence, user. Vous pouvez voir les liens qui existent entre ces modules dans l'annexe 6 qui contient un schéma des relations ontologiques.

2.1.2. L'environnement de développement intégré

NetBeans est un environnement de développement intégré libre. Ce qui rend NetBeans particulièrement intéressant pour développer Aniseed est qu'il supporte un grand nombre de langages de programmation, et qu'il est capable de travailler avec les systèmes de gestion de versions.

2.1.3. Le framework

L'usage d'un framework dans la version 4 va permettre de corriger de nombreux problèmes de la version 3. En effet, dans les précédentes versions d'Aniseed, il n'y avait pas d'organisation du code ce qui entrainait des difficultés pour maintenir et étendre le système.

L'implémentation d'un framework permet de :

- Faciliter la lecture du code
- Faire des tests facilement
- Ajouter de nouvelles fonctionnalités au site sans modifier tout le code du site
- Produire des modules réutilisables
- Corriger les bugs qui pourraient éventuellement apparaître (maintenance simplifiée)

Le framework utilisé pour développer la nouvelle version d'Aniseed est Jelix. Jelix est un framework open source proposant un ensemble d'interfaces de programmation : moteur de Template, accès aux données, gestion de droits... Jelix possède également plusieurs particularités : il permet le développement de modules réutilisables, qui sont des ensembles de fichiers concernant un domaine précis, et permet la création de composants jDao, ce qui permet de faire du mapping relationnel objet en ajoutant une couche d'abstraction de base de données.

Il permet donc aux développeurs de s'abstraire en partie des requêtes SQL en générant des objets PHP. Il opère en se basant sur des fichiers XML créés à l'aide d'un script fourni par Jelix. C'est dans ces fichiers qu'on peut définir la structure d'une table ainsi que les méthodes qui y sont associées. Ce fichier XML fournit deux objets différents : un objet record qui va permettre d'implémenter des insertions dans la base et une factory qui permet d'appeler les méthodes relatives à cet objet et donc par extension de récupérer les enregistrements pertinents.

La séparation logique du code obtenue grâce aux design patterns MVC et DAO* font de Jelix un framework complémentaire du schéma Chado.

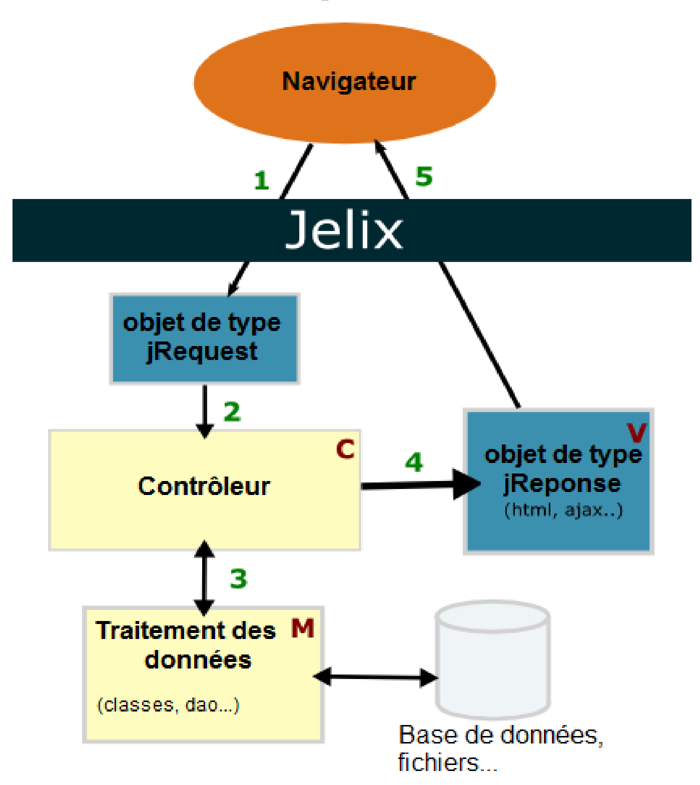


Figure 11 - Le fonctionnement de Jelix

Voici le principe de fonctionnement de Jelix (cf. figure 11):

1. Jelix reçoit du client une requête http. Il instancie un objet de type jRequest qui va contenir les données de la requête. Il instancie également le contrôleur lié à l'action.
2. La méthode du contrôleur correspondant à l'action est exécutée. Cette méthode récupère les paramètres éventuellement contenus dans la requête.
3. Le contrôleur va exécuter les traitements et éventuellement récupérer des résultats qui pourront être affichés.
4. Le contrôleur instancie un objet de type jResponse. Il initialisera les templates et assignera les données à afficher.
5. Jelix va récupérer cet objet jResponse et invoquer la génération du document en sortie et envoyer ce dernier au navigateur.

Jelix implémente le module junittest qui permet de réaliser des tests unitaires. Il utilise pour cela la bibliothèque SimpleTest. Cependant, je n'ai pas utilisé cette bibliothèque et j'ai seulement fait des tests fonctionnels (tests de validation) et de performance pour vérifier que l'application donnait bien le résultat attendu. La création de modules de tests était trop chronophage par rapport à la durée de mon stage, et de ce fait peu intéressante.

La syntaxe utilisée dans jTpl est à mi chemin entre celle utilisée dans le moteur de template Smarty*, et la syntaxe PHP. Le but étant d'avoir des templates suffisamment lisibles, facile à modifier en n'imposant pas une syntaxe trop éloignée de PHP, tout en proposant des facilités que ne possède pas PHP et propres à Jelix. Cette syntaxe présente l'avantage d'être très facile à prendre en main.

Enfin, Jelix possède une couche d'abstraction pour l'accès aux bases de données : jDb. Ce composant met à disposition de l'utilisateur le principe des transactions. Une transaction est une suite d'opérations effectuées sur une base de données. La suite d'opérations est indivisible, en cas d'échec en cours d'une des opérations, la suite d'opérations doit être complètement annulée quel que soit le nombre d'opérations déjà réussies.

Jelix présente aussi l'avantage d'être relativement rapide à prendre en main comparé à d'autres frameworks comme Symfony. Symfony a aussi le désavantage d'alourdir l'application en nombre d'octets.

2.1.4. Le serveur

Apache http Server est le serveur http libre le plus répandu sur Internet. C'est un logiciel connu pour sa performance et sa sécurité ainsi que pour sa modularité (encouragement de l'innovation et de la personnalisation).

2.2. L'ancienne version de l'interface de curation : analyse de son contenu et ses défauts

L'ancienne version du curateur présente, à part les problèmes du script et de la base de données, des problèmes de mise en page qui gênent la compréhension pour les utilisateurs. On trouve parfois des champs mal nommés ou des icônes dont le sens n'est pas toujours évident. Quelques améliorations sont également à faire pour augmenter la rapidité du travail du curateur. Sur la version 3, le curateur doit aller chercher sur le reste du site ou sur le site pubmed, des identifiants qui n'ont généralement aucun sens pour lui en temps que biologiste.

La nouvelle version devra proposer une version plus agréable pour l'utilisateur en mettant en place des systèmes d'auto complétion, de pop-up…

Il faut aussi savoir que le curateur se divise en quatre grandes sections. La première concerne les outils moléculaires, la seconde la description des profils d'expression de gènes et d'activité, la troisième la description des régions régulatrices, la quatrième la description des phénotypes.

Pour des raisons de contraintes de temps, mon stage se limite aux deux premières sections.

2.2.1. Outils moléculaires

Pour la page Create a Molecular tool (visible dans la figure 12), l'utilisateur devait saisir à la main le nom exact du gène dans le champ « associated transcript ». Un gène ayant plusieurs noms (appelés synonymes), il n'est pas évident quel nom doit être retenu. De plus, il n'est pas aisé de savoir quelle est la syntaxe exacte du gène (majuscule, minuscules…).

Le deuxième défaut de cette page est que l'identifiant de la publication devait être cherché sur le site pubmed. Pour optimiser le curateur, l'idéal serait de supprimer cette recherche coûteuse en temps en proposant par exemple une liste déroulante contenant l'ensemble des publications ayant trait au gène considéré.

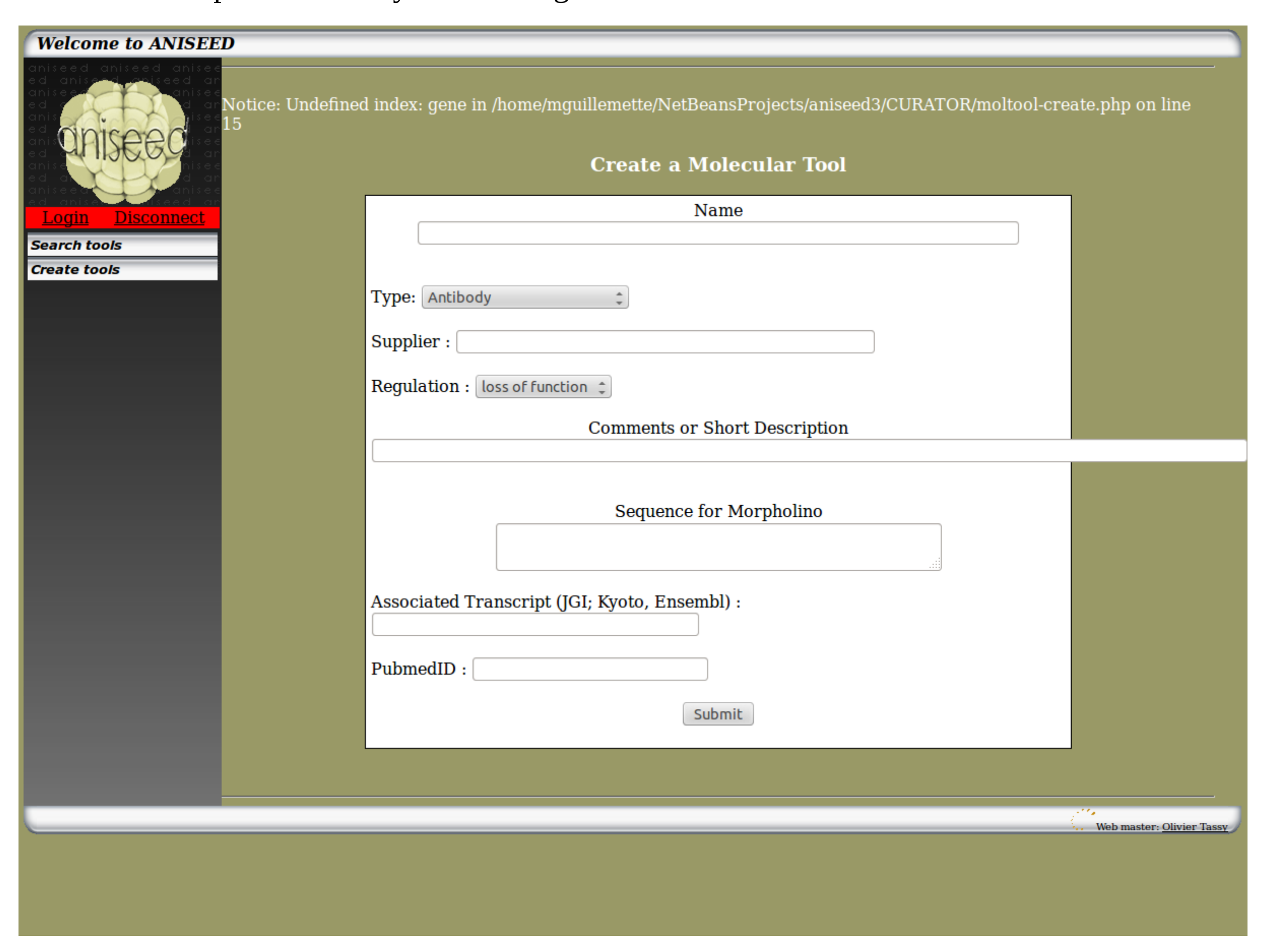


Figure 12 - Ancienne version de la page create a molecular tool

2.2.2. In situs

Pour la page Create an In situ data, la navigation manque de clarté. On précise sur la première page l'espèce, le stade de développement, l'auteur et si l'in situ est de type sauvage ou pas.

Sur la page suivante, on peut ajouter des images, des commentaires, des références associées à l'in situ, etc. Comme l'illustre la figure 13 la mise en page est étrange. On trouve plusieurs boutons « edit », mais il est difficile de savoir ce qu'ils permettent d'éditer…

Il manque également un objet à la définition d'un in situ : il n'existe pas le concept permettant de relier les profils d'expression de plusieurs gènes dans plusieurs conditions

expérimentales. Ce manque pose des problèmes car il y a certaines données qui ne sont de ce fait pas stockées dans la base.

Figure 13 - Ancienne version de la page create in situ

2.3. Mise en place des maquettes

2.3.1. Page de création d'un outil moléculaire

Suite aux problèmes évoqués, j'ai créé une maquette qui a ensuite subi plusieurs modifications. La maquette finale est visible ci-contre dans la figure 14.

Les champs Name, Type, Supplier, Regulation et Sequence n'ont pas subi de modification car ils n'en présentaient pas le besoin. Une nouvelle liste déroulante permettant de spécifier l'espèce a été mise en place. Cette liste permettra de restreindre la quantité de

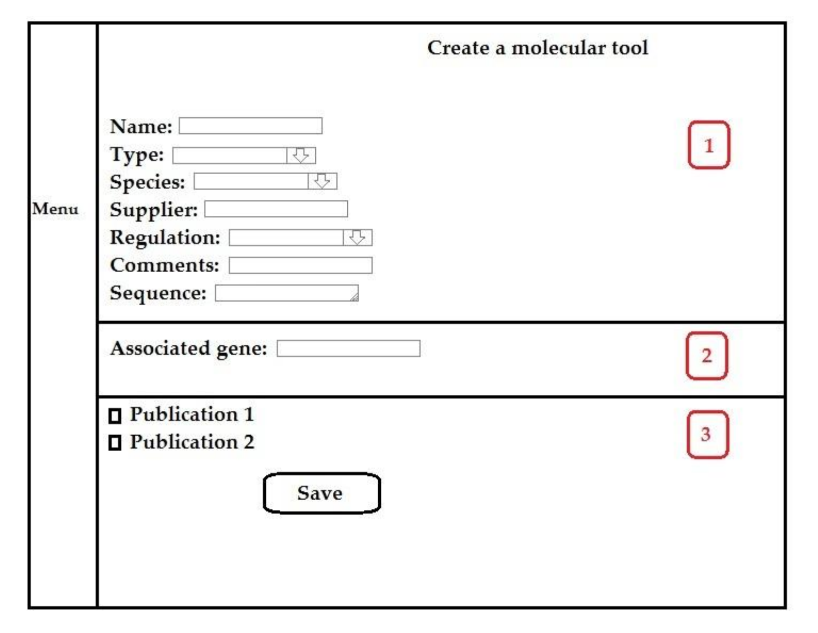


Figure 14 - Maquette de la page create a molecular tool

gènes proposés par l'auto complétion sur le champ « associated gene ». Lorsqu'un utilisateur saisira une espèce, le champ « associated gene » apparaîtra. L'utilisateur pourra ainsi saisir son gène (en s'aidant de l'auto complétion). L'auto complétion devra gérer le fait que l'utilisateur puisse saisir un synonyme (chaque gène a un identifiant et un nom commun, l'auto complétion fonctionnera avec ces deux termes). Il faudra également qu'il puisse commencer par le milieu du terme et non pas forcément le début.

Lorsque le champ « associated gene » sera rempli, une recherche s'effectuera pour trouver toutes les publications associées à ce gène. Vu le nombre restreint de publications associées à un gène (au plus une vingtaine de publications dans les organismes étudiés par l'équipe hôte) nous avons décidé de mettre une case à cocher par publication. Ainsi, si le gène a des publications qui lui sont associées dans la base de données, les cases à cocher apparaitront. Sinon, la phrase « no publications for this gene » apparaîtra. Dans les deux cas, il y aura le bouton de soumission qui apparaîtra également.

Le diagramme de séquences ci-dessous résume le déroulement de la création d'un outil moléculaire.

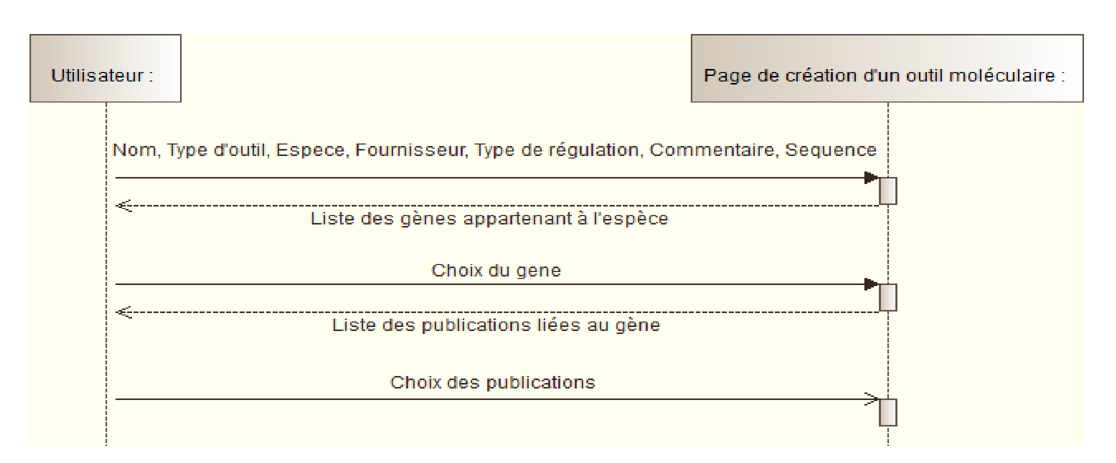


Figure 15 - Diagramme de séquences pour la création d'un outil moléculaire

2.3.2. Les pages de création d'une expérience de type in situ

Il y aura plusieurs pages pour permettre la création d'un in situ. Tout le processus de création d'une expérience de type in situ est résumé dans le diagramme de séquences visible dans l'annexe 4. Dans la première page, l'utilisateur créera son expérience et son biomatériel. (cf. figure 16 ci-contre)

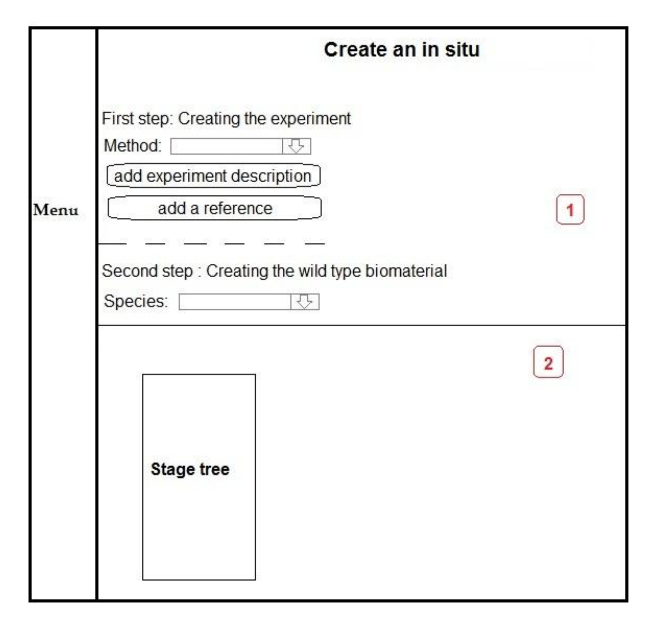


Figure 16 - Maquette de la page create in situ

La première section de la page s'affiche dés le chargement de la page. La section numérotée 2 s'affiche après sélection de l'espèce. Tous les champs visibles ici sont obligatoires. Les champs optionnels sont ceux s'affichant lors des

clics sur les deux boutons.

Lors du clic sur le bouton « add experiment description », un champ textarea apparaît. Il est ensuite possible de cacher le champ. De même, le bouton « add a reference » permet d'afficher un champ où l'utilisateur pourra saisir sa référence. (Ce champ bénéficie du phénomène d'auto complétion.) Il s'affichera également un autre bouton nommé « add more references ». Lors du clic sur ce bouton, un champ supplémentaire apparaîtra dans lequel l'utilisateur pourra insérer une publication supplémentaire. Vous pouvez cliquer autant de fois que nécessaire sur ce bouton.

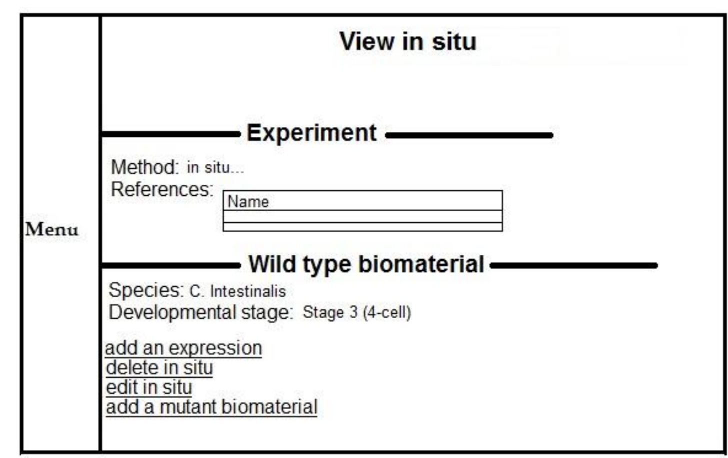


Figure 17 - Maquette de la page view in situ

Ensuite s'affichera la page faisant le résumé de l'expérience de type in situ complète. Comme dit dans le cahier des charges, une expérience de type in situ se compose de plusieurs éléments : une expérience, un biomatériel de type sauvage, 0 à n biomatériels de type mutant et pour chaque biomatériel, une expression. (cf. figure 5)

La figure 17 vous montre la page « view in situ » liée à une expérience de type in situ ne comportant pas encore d'expressions ou de biomatériels de type mutant. Si l'in situ comporte un biomatériel, les informations liées à celui-ci seront présentes sur cette page. De même pour une expression.

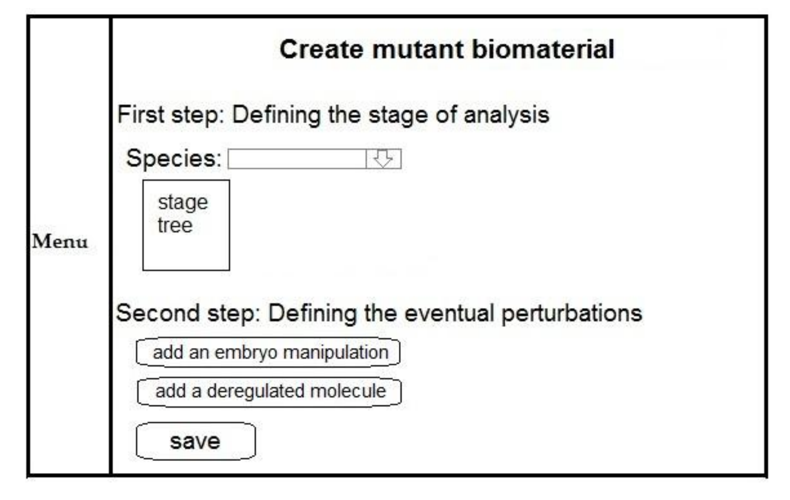


Figure 18 - Maquette de la page create biomaterial

Cette page permet donc de supprimer des éléments, ou d'accéder à la page pour les éditer. Cette page propose aussi des liens pour ajouter de nouveaux éléments. Enfin, elle permet de voir le résumé de tous les éléments liées à une expérience de type in situ.

Je vais maintenant vous présenter la page vous permettant d'ajouter un biomateriel de type mutant. Cette page comporte deux groupes de champs qui sont optionnels. C'est pourquoi l'utilisateur devra cliquer sur les boutons s'il veut y avoir accès.

En cliquant sur « add an embryo manipulation », l'utilisateur fera apparaître une liste déroulante où il pourra choisir le type de manipulation et le stade de développement de son embryon. Lorsqu'il aura choisi le stade de développement, il pourra choisir le territoire anatomique concerné. (La liste des territoires anatomiques dépend du stade de développement.)

En cliquant sur « add a deregulated molecule », l'utilisateur fera apparaître deux arbres avec les stades de développement (pour choisir le stade de départ et le stade de fin), et un champ où il pourra renseigner l'outil moléculaire. Le champ permettant de saisir son outil moléculaire bénéficiera d'une aide à l'auto complétion.

Enfin, je vais vous présenter la page permettant d'ajouter un profil d'expression. Que le profil d'expression soit lié à un biomatériel sauvage ou mutant, il faudra renseigner les mêmes informations donc la page est la même.

Grâce aux boutons radios situés en haut de la page, l'utilisateur pourra définir le niveau de visibilité et l'état de la curation. Lors du clic sur le bouton « add a probe », une fenêtre s'ouvrira. Dans cette fenêtre, l'utilisateur commencera par saisir le gène lié au clone qu'il veut associer à l'expression. Lorsqu'il aura saisi le gène (il pourra éventuellement se faire aider par un système d'auto complétion), la liste de clones liés à ce gène apparaîtra automatiquement. Il pourra ensuite choisir un de ces clones.

Lorsque l'utilisateur cliquera sur « add experimental conditions description », un champ texte apparaîtra.

Lorsqu'on clique sur "add territories of expression" une fenêtre s'ouvrira. Cette fenêtre contiendra un arbre anatomique où l'utilisateur pourra sélectionner plusieurs territoires à l'aide de cases à cocher. Pour chaque territoire anatomique, il pourra spécifier s'il est « part of », « not sure », ainsi que les positions subcellulaires touchées.

Les champs obligatoires seront signalés par une étoile.

Figure 19 - Maquette de la page create expression

2.4. L'agencement des pages

Comme vu précédemment l'utilisation de Jelix impose une répartition du code particulière. Le fonctionnement global est un contrôleur qui appelle un template. Dans le

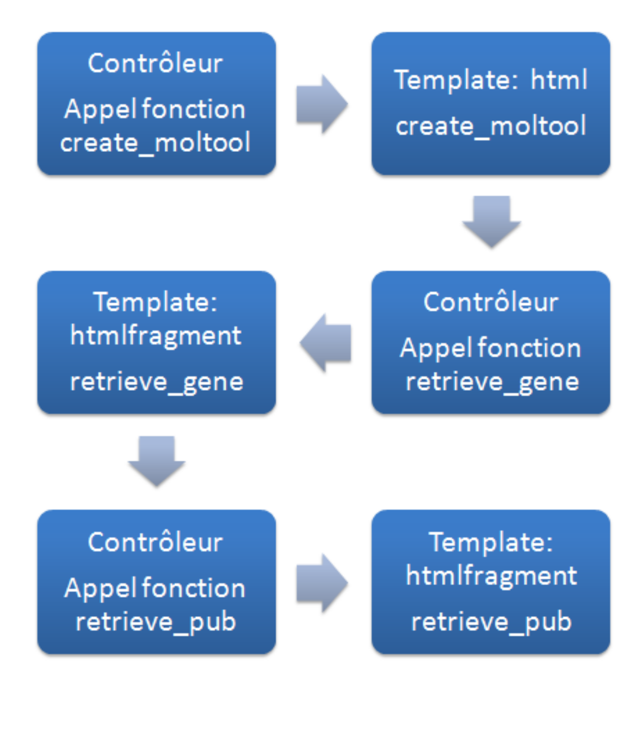


Figure 21 - Organisation du code

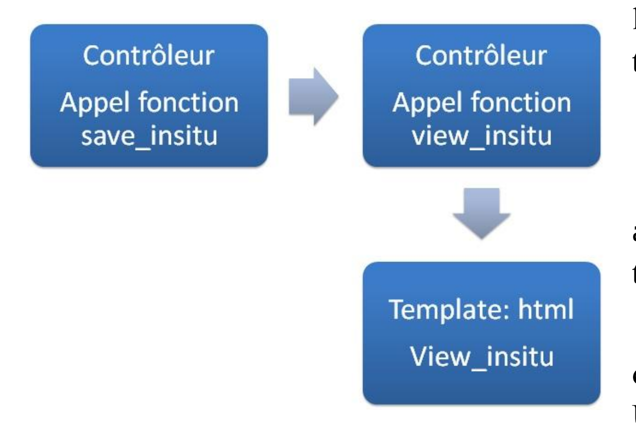


Figure 22 - Exemple de redirection

contrôleur, on peut faire appel à des classes pour créer des objets, et le template ne servira qu'à l'affichage des résultats.

Prenons l'exemple de la page « Create a molecular tool » pour expliciter l'organisation du code.

Pour charger la page contenant le formulaire utilisé pour créer un outil moléculaire, on commence par utiliser la fonction « create_moltool ». Cette fonction permet d'extraire des valeurs de la base de données (par exemple la liste d'espèces dont nous avons besoin), d'assigner ces valeurs au template, et enfin d'appeler le template voulu. Dans l'exemple choisi ici, une autre fonction sera appelée. Cette fonction aura pour but d'extraire de la base de données tous les gènes liés à l'espèce choisie, puis d'appeler un template de type htmlfragment (ce template ne contient que le code contenu dans le template, il n'y a pas d'inclusion du header, ni du menu…). Enfin, lorsque le gène est choisi, on appelle une autre fonction qui va récupérer toutes les publications liées à ce gène.

Un deuxième exemple va permettre d'illustrer le fonctionnement des redirections. Une redirection est utilisée lorsqu'on veut appliquer deux fonctions à la suite. Par exemple, après avoir enregistré l'expérience de type in situ dans la base, on veut rediriger l'utilisateur automatiquement vers la page pour voir le résumé des données concernant cet in situ. La réponse de la fonction « save insitu » sera de type redirect. On définira l'action de la réponse (c'est-à-dire la fonction vers laquelle on va la rediriger) ainsi que les paramètres à passer à cette fonction.

2.5. L'interaction avec la base de données

2.5.2. Le retrait d'informations de la BD

Le retrait d'informations de la base de données est simplifié par l'utilisation que fait Jelix des DAO*. jDAO est l'ORM* de Jelix : il permet de générer des objets PHP qui opèrent sur des tables précises de base de données. La génération de ces objets inclut la génération des requêtes SQL qui correspondent à ces opérations.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<dao xmlns="http://jelix.org/ns/dao/1.0">
    <datasources>
        <primarytable name="reagent" realname="nd_reagent" primarykey="nd_reagent_id" />
        <foreigntable name="cvterm" primarykey="cvterm_id" onforeignkey="type_id" />
        <optionalforeigntable name="feature" primarykey="feature_id" onforeignkey="feature_id" />
    </datasources>
    <record>
        <property name="nd_reagent_id" fieldname="nd_reagent_id" datatype="int" autoincrement="true" default="" />
        <property name="name" fieldname="name" datatype="varchar" required="true" maxlength="80" />
        <property name="type_id" fieldname="type_id" datatype="int" required="true" />
        <property name="feature_id" fieldname="feature_id" datatype="int" />
        <!--<property name="" fieldname="" datatype="string/int/float/date"
        required="yes" maxlength="" minlength="" regexp="" sequence=""
        updatepattern="" insertpattern="" selectpattern=""/>-->
        <property name="type_name" fieldname="name" datatype="string" table="cvterm" />
        <property name="feature_name" fieldname="uniquename" datatype="string" table="feature" />
    </record>
    <factory>
        <method name="getByUnique" type="selectfirst">
            <parameter name="name" />
            <parameter name="type_id" />
            <conditions logic="and">
                <eq property="name" expr="$name" />
                <eq property="type_id" expr="$type_id" />
            </conditions>
        </method>
        <method name="findFirstByName" type="selectfirst">
            <parameter name="name" />
            <conditions>
                <eq property="name" expr="$name" />
            </conditions>
        </method>
         <method name="findAllByPhenotype" type="php">
             <parameter name="phenotype_id" />
             <parameter name="start" default="0" />
             <parameter name="limit" default="-1" />
             <body><![CDATA[
                 if(empty($phenotype_id)) return null;

                 $sql  = $this->_selectClause;
                 $sql .= $this->_fromClause . ', phenotype_reagent';
                 $sql .= $this->_whereClause;
                 $sql .= ($this->_whereClause == '' ? ' WHERE ' : ' AND ') . 'phenotype_reagent.reagent_id = reagent.nd_reagent_id';


                 $sql .= ' AND phenotype_reagent.phenotype_id = '.$this->_prepareValue($phenotype_id, 'integer');

                 if($limit != 1) $sql .= ' ORDER BY reagent.name ASC';

                 $rs = $this->_conn->limitQuery($sql, $start, $limit);
                 $rs->setFetchMode(8, $this->_DaoRecordClassName);

                 return $rs;

             ]]></body>
         </method>
     </factory>
 </dao>
```

Figure 23 - Exemple de DAO

Pour ce faire, il faut fournir un ficher en XML (figure 23). Ce fichier va permettre au développeur de s’affranchir de l’écriture des requêtes SQL, des problèmes du genre SQL injection… À partir de ce fichier, jDAO va fournir deux objets : « record » représentant un enregistrement, ses propriétés correspondant avec les champs d’une table, et « factory » qui fournit un certain nombre de méthodes permettant de créer un record, de le sauvegarder dans la base, de le supprimer, ou de récupérer des ensembles d’enregistrements.

2.5.3. Les enregistrements dans la BD

Pour ajouter un enregistrement à une table, on va de nouveau utiliser une factory. On commence par appeler la factory ciblée, puis on crée un enregistrement vide du même type que la table concernant la factory. On complète ensuite cet enregistrement en attribuant une valeur à chaque champ de la table.

Dans l’exemple de la figure 24, on commence par récupérer le schéma dans lequel aura lieu l’enregistrement. Ensuite, on appelle l’objet jDAO, et on crée un enregistrement du

même type. On remplit ensuite cet enregistrement avant de finalement l'insérer. S'il y a des champs de type autoincrement, les propriétés correspondantes dans $moltool seront mises à jour avec la valeur.

```
$schema = jClasses::getService('organism~organismService')->getSchema(jDao::get('organism~organism')->get($Specie));

//Saving data concerning the reagent
$moltool_factory = jDao::get('nd~reagent', 'manager');
$moltool = jDao::createRecord('nd~reagent');
$moltool->name = $Name;
$moltool->type_id = $Type;
$moltool_factory->insert($moltool, $schema);
```

Figure 24 - Enregistrement dans la base de données

Dans les annexes 2 et 3 vous trouverez des diagrammes de classes propres aux données que j'ai traitées. Pour des soucis de clarté, je n'ai inclus dans le diagramme de classes des in situs que les attributs et classes qui sont nécessaires à votre compréhension.

Un diagramme de classes général de l'application est visible dans l'annexe 5.

2.5.4. L'édition et la suppression dans la BD

Pour éditer une information contenue dans la base de données, il faut d'abord récupérer l'enregistrement correspondant (cf. figure 25).

```
//Updating the informations
//Data concerning the reagent
 $moltool_factory = jDao::get('nd~reagent', "manager");
 $record = $moltool_factory->get($id);
 $record->name = $Name;
 $record->type_id = $Type;
 $moltool_factory->update($record);
```

Figure 25 - Mise à jour dans la base de données

La méthode « get » créée automatiquement pour toutes les DAO permet de récupérer l'enregistrement avec la clé primaire. Une fois l'enregistrement récupéré, on peut modifier les données qu'il contient puis le mettre à jour dans la base grâce à la fonction update().

Pour la suppression d'un enregistrement contenu dans la base de données, on récupère la factory concernée. Puis on supprime l'enregistrement en appelant la méthode delete() sur la factory en précisant la clé primaire de l'enregistrement en paramètres.

```
//Deleting the data from the database
    //Information about the reagent
    $moltool_factory = jDao::get('nd~reagent', "manager");
    $moltool_factory->delete($id);
```

Figure 26 - Suppression dans la base de données

2.6. L'utilisation de Jquery et Jquery UI

2.6.1. Les listes liées

Lorsque je suis arrivée au CNRS, je ne connaissais du web que les langages suivants : html, PHP et SQL. Je savais qu'il était possible de créer des listes liées car j'avais déjà eu l'occasion de voir ce système sur d'autres sites, mais j'ignorais comment réaliser cela. Il a donc fallu que je fasse de nombreuses recherches sur internet pour savoir quel langage ou outils il faudrait utiliser. J'ai ensuite fait quelques tentatives peu fructueuses en utilisant Ajax. Peu après, mon tuteur m'a présenté un sélecteur de Jquery nommé change[4]. Ce sélecteur permet de provoquer un événement lors du changement d'un champ.

Son fonctionnement global est visible dans la figure 27 ci-dessous.

```
92   $("#organism_id").change(retrieve_genes);
93
94   function retrieve_genes(){
95       $.ajax({
96           url: "{/literal}{jurl 'curator~moltool:retrieve_gene'}{literal}",
97           data: $(":input").serializeArray(),
98           success: function(result){
99               $("#result_gene").html(result);
100          }
101      });
102      return false;
103  }
104
```

Figure 27 - Sélecteur change

Dans la ligne 92, on précise que pour la liste déroulante dont l'id est « organism_id », lors d'un changement de sélection, on appelle la fonction retrieve_genes.

La fonction retrieve_genes est ensuite explicitée ligne 94. Elle appelle un url et envoie tous les champs du formulaire (:input) en paramètre à cette URL sous forme de tableau (serializeArray). La réponse de cette page appelée sera ensuite placée dans la balise d'id « result_gene ».

L'appel visible ligne 96 va en fait appeler la fonction retrieve_pub qui est située dans le module curator, dans le contrôleur moltool. Cette fonction, visible dans la figure 28 va renvoyer une réponse de type htmlfragment car elle ne renvoie qu'une portion de page et non une page entière. En effet, le type jResponseHtml aurait contenu le <head>, le <body>…

Le reste de la fonction est assez classique, ligne 229 on récupère les valeurs intéressantes dans la base de données puis on lie la variable $pubs au Template. On désigne ensuite quel est le template associé. On fait ensuite l'affichage de toutes ces publications dans le template.

[4] http://api.jquery.com/change/

```
function retrieve_pub(){
    jClasses::inc('pub~pub');
    $resp = $this->getResponse('htmlfragment');

   //Recovering the information from the template
    $gene = $this->param('gene_id');
    $id = $this->param('id');

   //Recovering informations concerning the pub that was selected before
    $pubrecord_factory = jDao::get('nd~reagent_pub');
    $rpubs = $pubrecord_factory->getByReagent($id);
    $tb = $rpubs->fetchAll();
    //$tb is an array containing all the publications that were selected
    $resp->tpl->assign('tb_old_pub', $tb);

    //Retrieving the information concerning all the publications in the database
    $pubs = jClasses::getService('pub~pubService')->findAllByFeature($gene);
    $resp->tpl->assign('Pubs', $pubs);

    $resp->tplname = 'curator~retrieve_pub';

   return $resp;
}
```

Figure 28 - Fonction retrieve_pub

2.6.2. Auto complétion

L'auto complétion de certains champs était impérative. En effet, les curateurs perdent trop de temps dans la version 3 pour chercher la syntaxe exacte des gènes, les identifiants de publication, etc.

Dans un premier temps j'ai cherché le moyen le plus simple possible pour permettre cette auto complétion. J'ai donc appris comment proposer de l'auto complétion en utilisant seulement HTML5. Sur la figure 29 est présenté le code permettant cette auto complétion. Il y aura ensuite une explication plus détaillée de ce code.

```
<input name="q" type="text" list="liste_valeurs"> ①
<datalist id="liste valeurs"> ②
<option label="suggestion 1" value="suggestion 1"/>③
<option label="suggestion 2" value="suggestion 2"/>
<option label="suggestion 3" value="suggestion 3"/>
...
</datalist>
```

Figure 29 - Autocomplete avec HTML5

① Cette première ligne de code permet de créer une zone de saisie de texte libre. C'est dans ce champ que l'utilisateur commencera à insérer sa valeur. L'attribut « list » sert à définir l'id du datalist qui contiendra tous les mots à reconnaitre.

② La datalist va comporter toute la série de valeurs qui seront proposées pendant la saisie. Dans cette datalist, on a pour chaque valeur une balise option.

③ Pour chaque option, on trouve le « label » qui correspond au libellé affiché comme suggestion lors de la frappe ainsi que la « value » qui sera la valeur réellement retenue et envoyée lors de la validation du formulaire.

Notons que cette fonctionnalité d'auto complétion ne peut pas être utilisée dans notre cas pour plusieurs raisons:

- HTML5 est supporté par la plupart des versions récentes des navigateurs classiques. Or, les utilisateurs du curateur ne font pas forcément les mises à jour de leur navigateur.
- Même sur certains navigateurs comme Opera où la fonctionnalité est bien intégrée, on ne peut trouver un terme qu'en tapant le début du mot. (Par exemple si on recherche le gène « Trypsin type protease 26 » il faudra absolument taper « Trypsin » pour trouver le gène. Avec « protease », le gène voulu ne sera pas dans les résultats.) Dans notre cas, il faut absolument pouvoir saisir le milieu du terme voulu pour le trouver.
- Il est hors de question d'intégrer dans la page stockant le formulaire l'intégralité de la liste des gènes. La fonctionnalité d'auto complétion proposée par HTML5 est donc réservée aux listes comprenant peu de valeurs.

De par ces multiples raisons, il faut absolument utiliser une autre méthode pour l'auto complétion du curateur. C'est pourquoi je me suis tournée vers Jquery UI qui propose un widget d'auto complétion[5] assez facilement implémentable. L'utilisation que j'ai faite de ce widget est décrite dans les figures ci-dessous (figures 30 et 31).

Dans un premier temps il s'agit de récupérer la liste de valeurs. Cette opération se fait dans le contrôleur. On commence par récupérer tous les gènes dans la base de données, cette opération est visible ligne 271. (On ne prend que les gènes liés à l'espèce choisie auparavant dans la page.)

```
265 //Retrieving the information concerning all the genes in the database
266      $genesByOrganism_cacheName = 'genesByOrganism__organismId_'.$Specie;
267
268      $tb = jCache::get($genesByOrganism_cacheName); //will contain all the gene names
269      if (empty($tb)){
270         $genetype = jDao::get("cv~cvterm")->getByUnique("sequence", "gene");
271         $gene = jDao::get("sequence~feature")->findAllByOrganismAndType($Specie, $genetype->cvterm_id);
272         foreach($gene as $g){
273             $Geneobj = new Gene($g);
274             $Geneobj->setGeneNames();
275             $tb[] = $Geneobj;
276         }
277         jCache::set($genesByOrganism_cacheName, $tb);
278      }
279
280     $resp->tpl->assign('TbGene', $tb);  //Synonym_name
281     $resp->tplname = 'curator~retrieve_gene';
282     $cnx->commit();
283
284     return $resp;
285 }
```

Figure 30 - Contrôleur retrieve_gene

Lorsqu'on a cette liste de gènes, on va la transmettre de façon classique au template à travers une variable.

[5] http://api.jqueryui.com/autocomplete

```
3  <table>
4      <tr>
           <td>Associated gene</td>
6          <td><input type="text" name="gene_id" id="gene_id" autocomplete = "off" value="{$gene_id}">
7         {if $id != null} <td>The name of the gene that was associated to this molecular tool is {$gene_name}</td>{/if}
8      </tr>
9  </table>
10
11 <script type="text/javascript">
12 {literal}
13     var liste_gene = [
14 {/literal}
15     {foreach $TbGene as $g}
16     {literal}{ label: "{/literal}{=addslashes($g->feature->name)} ( {$g->getGeneNames()} ) {literal}", value: "{/literal}{$g->feature->feature_id}{literal}"},{/literal}
17 {/foreach}
18 {literal}
19 ];
20
21 $('#gene_id').autocomplete({
22     source : liste_gene,
23     delay: 500
24 });
25
26 {/literal}
27 </script>
```

Figure 31 - Template retrieve_gene

Dans le template (cf. figure 31 visible ci-dessus), on crée un champ de type texte. L’auto complétion de la balise doit être mis à off pour que l’utilisateur n’ait pas l’auto complétion automatique relative à son historique.

Ensuite, on crée une variable javascript (ligne 13) qui contiendra toutes nos valeurs. Ici, je n’ai pas utilisé une simple liste de valeurs car si l’utilisateur tape bien le nom du gène, je veux surtout pouvoir récupérer l’identifiant. Ainsi, le label correspond aux valeurs qui s’afficheront pour aider la saisie alors que la value contiendra l’identifiant du gène choisi. Ainsi, lorsque la personne clique sur la valeur de son choix dans la liste d’auto complétion, c’est l’identifiant qui s’affiche dans le champ.

Ensuite, il convient de préciser pour quel champ on veut appliquer cette liste (ligne 21). On indique donc qu’on veut appliquer le widget d’auto complétion au champ dont l’id est « gene_id ». On précise également la source qui est la variable décrite au dessus, puis, le délai (cette information est optionnelle). Le délai permet de définir au bout de combien de secondes on va faire des suggestions à l’utilisateur.

2.6.2. Show/Hide

Pour augmenter la rapidité du travail du curateur, il a été décidé de pouvoir afficher et cacher les champs optionnels. Cette fonctionnalité a aussi été mise en place grâce à Jquery UI. En effet, Jquery UI propose deux méthodes simples à utiliser nommées « show » et « hide ».

Vous trouverez un exemple de l’utilisation de hide[6] et show[7] dans les figures ci-dessous.

[6] http://api.jqueryui.com/show/
[7] http://api.jqueryui.com/hide/

```
87  <tr>
88      <td>
89          <button type="button" id="Addcomm_exp">Add experimental conditions description</button>
90          <br>
91          <td>
92              <div id="hidden_addcomm_exp" style="display: none">
93                  <button type="button" id="Hide_comm_exp">Hide</button>
94                  <br>
95                  <textarea name="comm_exp" rows="5" cols="40" >{$Insitu->wild_type->expression->expressi
96              </div>
97          </td>
98      </td>
99  </tr>
```

Figure 32 – Exemple de Hide et Show (Template)

Dans ce morceau de code, on peut voir le bouton permettant d'ajouter une description. Ce bouton va en fait permettre d'afficher tout ce qui est contenu dans la div située en dessous de lui. Ce mécanisme va être mis en place grâce au morceau de JavaScript ci-dessous.

```
//Experimental conditions
$("#Addcomm_exp").click(function () {
        $("#hidden_addcomm_exp").show("slide")});
$("#Hide_comm_exp").click(function () {
        $("#hidden_addcomm_exp").hide("slide")});
```

Figure 33 - Exemple de Hide et Show (JavaScript)

Au chargement de la page, la div ligne 92 est cachée. Lors du clic sur le bouton, tout son contenu va s'afficher. Enfin, lors du clic sur le bouton hide (ligne 93), la div disparaît à nouveau.

2.6.3. Clone

Il a parfois été nécessaire de créer un moyen pour que l'utilisateur puisse dupliquer des champs. Par exemple, pour associer autant de références que nécessaire à son expérience, j'ai mis en place des champs qui se clonent avec du JavaScript.

```
//Add more references
$(".addRef").click(function (){
    var form = $(this).closest('form');
    var refList = form.find('.clonedInputRef');
    // n is the number of fields for references already present on the page
    var n = refList.length;
    // firstref is the first reference field present on the page
    var firstref = $(refList[0]);
    // lastref is the last reference field present on the page
    var lastref = $(refList[n-1]);
    // clonedref is the new cloned field
    var clonedref = firstref.clone();

    // For every input field that is cloned
    clonedref.find(':input').each(function() {
        // Emptying the field
        $(this)
            .filter(':text').val('').end()
    })

    // Adding the new field after the last field that was present
    clonedref.insertAfter(lastref).hide().fadeIn('slow');
});
```

Figure 34 - Clone

Dans la figure 34, on voit que lorsque l'utilisateur clique sur le bouton de classe « addRef », une fonction s'exécute. Cette fonction va commencer par rechercher dans le formulaire le plus proche tous les champs de classe « clonedInputRef », et les stocker dans la variable nommée refList. On va ensuite compter le nombre de références qui ont été crées, c'est-à-dire compter le nombre d'éléments dans la liste précédemment créée.

Aux lignes 113 et 115, on va récupérer respectivement le premier élément de la liste, et le dernier. Ensuite, ligne 117, on clone le premier élément afin de produire le nouveau champ. Ligne 120 on va vider le contenu du champ qu'on duplique.

Enfin, ligne 127, on insère le nouveau champ après tous les autres champs de la page.

2.6.4. Dialog

Le widget Dialog[8] mis en place par Jquery UI permet de faire apparaître des pop-ups qui peuvent être de simples messages, des formulaires… M'étant servie à plusieurs reprises de ce widget, j'ai décidé de vous présenter son fonctionnement à travers un cas simple.

```
25      <tr>
26          <td>
27              <div class="clone-form" title="Adding probes to your wild type expression">
28                  <p>First step: please enter a gene </p>
29                  <div>
30                      <input type="text" name="gene_id" id="gene_id" value="{$old_gene}" autocomplete="off"/>
31                      <p id="clones">
32                  </div>
33               </div>
34               <p id="summary_clones"></p>
35          </td>
36      </tr>
37      <tr>
38          <td><button type="button" id="add_probe" class="add_clone">{if $expression_id != null}Change {else} Add a {/if} probe</button></td>
39      </tr>
```

Figure 35 - Code HTML nécessaire à la création d'un Dialog

La figure 35 visible ci-dessus présente l'utilisation du dialog dans le template. Il faut créer une div englobant le contenu que l'on veut avoir dans sa fenêtre. (Ici c'est la div ligne 27). Il faut aussi créer le bouton entraînant l'ouverture de la fenêtre (ligne 38).

Pour mettre en place ce widget au niveau JavaScript, on commence par définir les propriétés de la fenêtre. AutoOpen permet de définir si la fenêtre s'ouvre automatiquement au chargement de la page. Modal permet de définir si les éléments à l'arrière de la fenêtre restent cliquables et actifs. Ensuite, on précise quels boutons seront présents dans la fenêtre.

Lorsqu'on clique sur le bouton « submit », plusieurs actions vont s'exécuter. Dans un premier temps, on va récupérer la case que l'utilisateur aura cochée. On va placer la valeur de cette case dans un champ caché sur la page. Ensuite, on cache le bouton « add probe » et on le remplace par le bouton « change probe ». Si l'utilisateur clique sur « change probe » alors la valeur du champ cachée prendra la nouvelle valeur. Enfin, on peut fermer la fenêtre.

---

[8] http://api.jqueryui.com/dialog

```
//Clones
//dialog for clones
$( ".clone-form" ).dialog({
autoOpen: false,
height: '850',
width: '850',
modal: true,
buttons: {
    "Submit": function() {
        var Checked = $('input[name="clone_id_selected"]:checked');
            $( "input#clone_id" ).val(Checked.value);
            $( "#change_probe" ).show();
            $( "#add_probe" ).hide();
        $( this ).dialog( "close" );
        $( "#old_probe" ).hide();
     },

    Cancel: function() {
    $( this ).dialog( "close" );
    }
    }
});

$(".add_clone" ).button().click(function() {
    $( ".clone-form" ).dialog( "open" );
});
```

Figure 36 - Dialog

2.7. La mise en cache

La mise en cache permise par Jelix a été très utile car certaines requêtes ou création d'objets nécessitait beaucoup de temps. De ce fait, il était utile de pouvoir créer un fichier temporaire contenant ces résultats plutôt que de refaire l'opération à chaque chargement de la page.

```
265 //Retrieving the information concerning all the genes in the database
266     $genesByOrganism_cacheName = 'genesByOrganism__organismId_'.$Specie;
267
268     $tb = jCache::get($genesByOrganism_cacheName); //will contain all the gene names
269     if (empty($tb)){
270         $genetype = jDao::get("cv~cvterm")->getByUnique("sequence", "gene");
271         $gene = jDao::get("sequence~feature")->findAllByOrganismAndType($Specie, $genetype->cvterm_id);
272         foreach($gene as $g){
273             $Geneobj = new Gene($g);
274             $Geneobj->setGeneNames();
275             $tb[] = $Geneobj;
276         }
277         jCache::set($genesByOrganism_cacheName, $tb);
278     }
279
280    $resp->tpl->assign('TbGene', $tb);  //Synonym_name
281    $resp->tplname = 'curator~retrieve_gene';
282    $cnx->commit();
283
284    return $resp;
285 }
```

Figure 37 - Mise en cache

La figure 37 montre la procédure à faire pour stocker des objets dans un fichier temporaire. On commence par définir le nom qu'on va attribuer au fichier stocké en cache (ligne 266). Ce nommage contient le nom de l'espèce afin qu'il y ait un fichier temporaire par espèce. Si le fichier en cache portant ce nom n'existe pas déjà, alors on va le créer en y insérant la variable qui nous intéresse. De ce fait, l'utilisateur ne créera qu'une fois ce fichier, les autres fois le navigateur récupèrera le contenu stocké en cache sur le serveur.

2.8. Jstree

Jstree[9] est un plugin libre se basant sur jquery. Sa documentation est en anglais. Ce plugin permet de créer des arbres semblables à ceux visible dans la figure 38 visible ci-dessous.

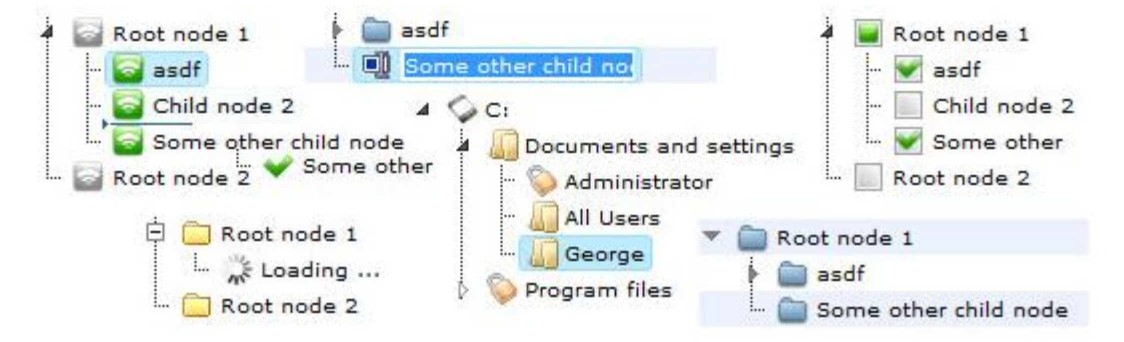


Figure 38 – Plugin Jstree, aperçu des fonctionnalités

Ce plugin a été utilisé de nombreuses fois sur le curateur, généralement même plusieurs fois par page. Nous allons nous pencher sur un des cas où a été utilisé cet arbre afin d'illustrer son fonctionnement général.

Le contrôleur permettant la création de l'arbre est visible dans la figure 40.

Dans un premier temps, on commence par inclure les classes dont on a besoin. La classe anatomyInsituJTreeRecursiveIterator dérive de la classe RecursiveIterator qui dérive elle-même de la classe Iterator. Ensuite, il faudra récupérer tous les territoires anatomiques liés au stade de développement sélectionné. C'est ce que fait la fonction findAnats() définie dans la classe devstage. On va ensuite chercher l'arbre qui est composé de ces territoires. Il faudra également assigner $it, l'itérateur de l'arbre, au template. Cet itérateur est un objet permettant de parcourir tous les éléments contenus dans l'arbre. Le parcours qu'il fait est décrit dans la classe Iterator et RecursiveIterator.

```
//Displaying anatomy tree
 jClasses::inc('aniseed~jtree/JTreeIterator');
 jClasses::inc('aniseed~jtree/anatomyInsituJTreeRecursiveIterator');

 $devstage = jClasses::getService('anatomy~devstageService')->get($devstage_id);
 $devstage->findAnats();

 $tree = jClasses::getService('anatomy~anatomyService')->getTree($devstage->anats);
 $it = new anatomyInsituJTreeRecursiveIterator($tree, new JTreeIterator($tree->getTree()), true);

 $tpl->assignByRef('tree_it', $it);


 $resp->body->assign('MAIN', $tpl->fetch('create_expression'));
 return $resp;
```

Figure 39 - Création d'un arbre dans le contrôleur

La figure 40 illustre l'affichage de l'arbre dans le template.

Les boutons permettent de faire ouvrir l'arbre entièrement afin d'afficher toutes les feuilles, ou de fermer l'arbre complètement afin de ne voir que les nœuds.

[9] www.jstree.com

```
<tr>
     <div class="anat-form" title="Territories of expression">
         <p>Please select the anatomical part(s) that are stained. </p>
         <div>
            <div class="buttons">
                <a id="close_all_anat"><img src='{$j_themepath}img/icons/contract.png' border=
                <a id="open_all_anat"><img src='{$j_themepath}img/icons/expand.png' border='0'
            </div>
            <br><br><br>
            <fieldset>
                <div id="anat_tree">
                        {foreach $tree_it as $it}
                        {/foreach}
                        {$tree_it->getStr()}
                </div>
            </fieldset>
         </div>
     </div>
</tr>
```

Figure 40 – Affichage de l'arbre dans le template

Enfin, un peu de JavaScript est nécessaire pour l'affichage de l'arbre et le fonctionnement des boutons.

```
//anatomy tree
    $(function () {
    $("#open_all_anat").click(function () {
        $("#anat_tree").jstree("open_all");
    });
    $("#close_all_anat").click(function () {
        $("#anat_tree").jstree("close_all");
    });
    $("#anat_tree").jstree({
            "ui" : {
                "select_limit" : 1,
                "initially_select" : [ "node_{/literal}{$anatomy_id}{literal}" ]
            },

            "types" : {
                "valid_children" : [ "root" ],
                "type_attr" : "title",
                "types" : {
                    "organism" : {
                        "icon" : {
                            "image" : "{/literal}{$j_themepath}{literal}img/icons/tadpole.png"
                        }
                    },
                    "cell" : {
                        "icon" : {
                            "image" : "{/literal}{$j_themepath}{literal}img/icons/cell.png"
                        }
                    },
                    "default" : {
                        "icon" : {
                            "image" : "{/literal}{$j_themepath}{literal}img/icons/region.png"
                        }
                    }
                }
            },
            "plugins" : [ "themes", "html_data", "ui", "types" ]
        });
    });

$(".add_anat" ).button().click(function() {
    $( ".anat-form" ).dialog( "open" );
});
```

Figure 41 - JavaScript pour l'affichage des arbres

On commence par utiliser le plugin ui[10]. Ce plugin permet de régler les sélections et désélections. Comme vous pouvez le constater ligne 285, on a imposé que l'utilisateur ne puisse choisir qu'un territoire anatomique. Ligne 286, on précise qu'un nœud est présélectionné.

On utilise ensuite le plugin types[11] qui permet de mettre en place le typage des nœuds. Ainsi, on pourra préciser le type de nœud valide (ligne 290), l'attribut de chaque nœud de liste (ligne 291), les différents types de nœud (par exemple ligne 295, 300 et 305 c'est l'icone qui les représente qui ont été modifiés)…

2.9. Subversion

Subversion, aussi couramment appelé « svn », est un logiciel de gestion de versions. En le combinant à l'utilisation de NetBeans, ce logiciel m'a permis de récupérer le projet Jelix correspondant à la version en cours de développement d'Aniseed et d'y apporter mes propres modifications en local. Le pouvoir de ce logiciel réside dans sa capacité à comparer plusieurs versions des différents fichiers. Ainsi, lorsque j'avais fini de développer ou d'améliorer une partie de mon code, j'ai pu proposer les différents fichiers que j'avais créés/édités, sans altérer le contenu de ces derniers s'ils ont été utilisés entre temps par un autre développeur.

Tout au long de mon stage, j'ai dû mettre à jour régulièrement ma copie locale. Les mises à jour ont été facilitées par NetBeans. Ces mises à jour ont toutes été répertoriées sur Redmine.

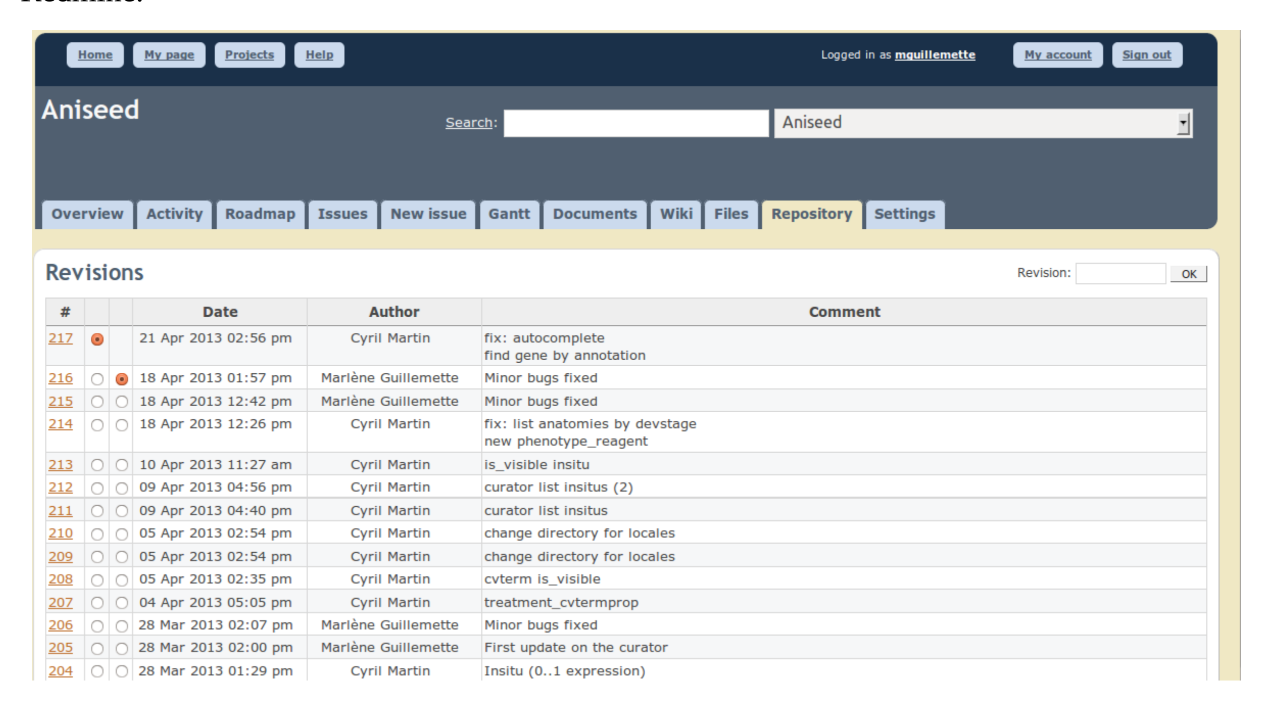


Figure 42 - Aperçu SVN

[10] http://www.jstree.com/documentation/ui
[11] http://www.jstree.com/documentation/types

3. Manuel d'utilisation

3.1. Les outils moléculaires

Aniseed supporte la description de profils d'expressions et de phénotypes en réponse aux perturbations de la fonction d'un gène. Pour définir ces perturbations, il utilise le concept d'outil moléculaire. Un outil moléculaire peut être un morpholino, un mRNA… Chaque outil moléculaire sera lié au gène qu'il régule (en lui apportant un gain ou une perte de fonction) ainsi qu'aux articles et expériences dans lesquels il a été décrit ou utilisé.

3.1.1. Ajouter un outil moléculaire

Pour ajouter un outil moléculaire, sélectionnez dans le menu situé à gauche de la page la section « create tools ». Ensuite cliquez sur « Create a Molecular tool ». La page de création d'un outil moléculaire s'affichera (figure 43).

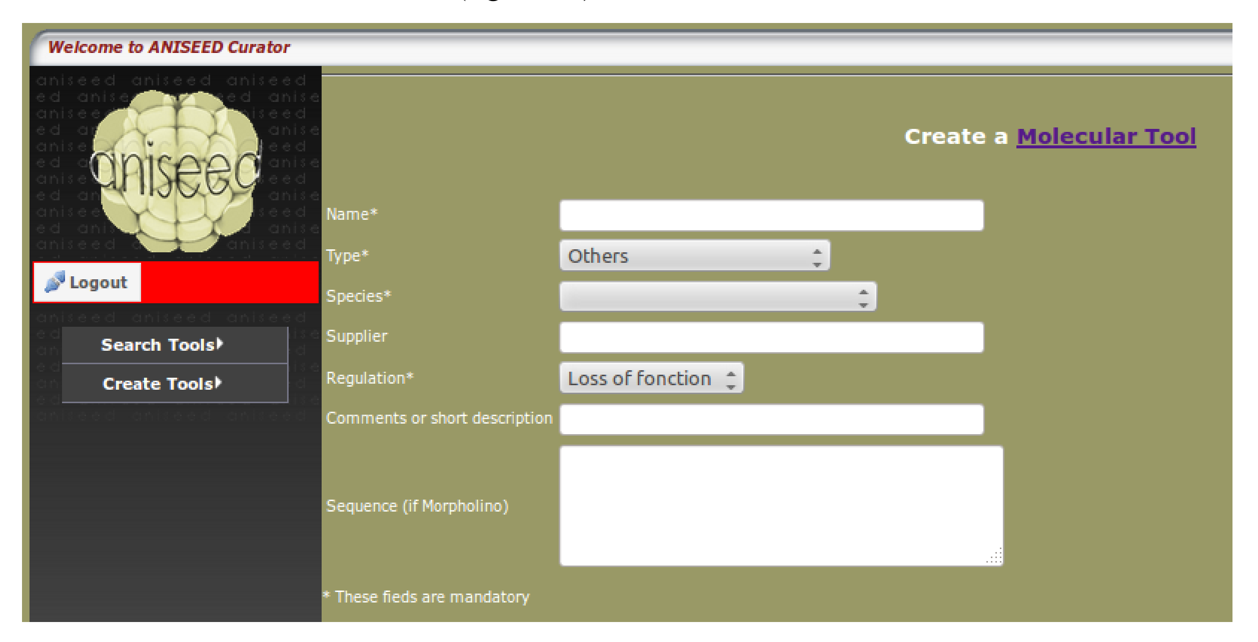


Figure 43 - Ajouter un outil moléculaire

Les champs obligatoires sont signalés par une étoile à côté de leur nom.

- Pour commencer, renseigner le nom de l'outil moléculaire. Si c'est un morpholino, spécifiez la personne qui l'a créé ou le laboratoire d'origine. (Exemple : FGF9/16/20-MO (Lemaire Lab))
- Choisissez le type de l'outil moléculaire en naviguant dans la liste déroulante. L'outil moléculaire peut être de cinq types différents : Inhibitor, overexpression construct, morpholino oligo, mRNA ou Others (qui regroupe tout autre type).
- Spécifier ensuite l'espèce possédant le gène associé à l'outil moléculaire. Lorsque vous aurez choisi votre espèce un champ supplémentaire s'ajoutera automatiquement dans le formulaire.
- Saisir ensuite dans le champ « Supplier » le fournisseur de l'outil moléculaire, si celui-ci est connu.
- Spécifier ensuite le type de régulation grâce à une liste déroulante. Il sera soit un gain de fonction soit une perte de fonction.

✦ Ajouter un commentaire ou une courte description de votre outil moléculaire dans le champ dédié à cet effet.

✦ Saisissez la « sequence ». Cette information est peu utile pour certains types d'outils moléculaires mais peut parfois être cruciale par exemple pour un morpholino.

✦ Vous pouvez ensuite inscrire dans le champ « associated transcript » le gène lié à votre outil moléculaire. Que vous saisissiez le nom exact du gène ou un de ses synonymes n'a pas d'importance puisqu'il y a une aide à l'auto complétion. Par exemple, si vous saisissez « Otx » l'auto complétion vous proposera KH.C4.84 puisque c'est l'identifiant du gène.

✦ Lorsque vous aurez saisi le gène, soit quelques cases à cocher apparaîtront, soit la phrase « No publications for this gene ». Dans le premier cas, vous pouvez cocher les publications où l'outil moléculaire est cité, puis enregistrer l'outil. Dans le second cas, vous pouvez directement enregistrer l'outil.

3.1.2. Voir la liste des outils moléculaires



Figure 44 - Lister les outils moléculaires

Si vous avez besoin de consulter la liste complète des outils moléculaires, il vous faudra sélectionner dans le menu situé à gauche de la page la section « search tools ». Ensuite cliquez sur « Molecular tools ». La page listant tous les outils moléculaires présents dans la base s'affichera (figure 43).

Cette liste vous permet de voir le nom de tous les outils ainsi que leur type. Ils sont d'office classés par ordre chronologique d'entrée dans la base (le plus ancien est en haut de la liste, le plus récent en bas). Si vous souhaitez afficher plus de détails sur un outil moléculaire, cliquez sur « edit » à côté de l'outil moléculaire choisi. Vous pourrez ainsi voir toutes les informations sur votre outil moléculaire.

3.1.3. Supprimer un outil moléculaire

Pour supprimer un outil moléculaire via l'interface de curation, il faudra que vous trouviez d'abord votre outil dans la liste des outils moléculaires. Cette liste est accessible via le menu situé à gauche de la page dans la section « search tools ». Cliquez ensuite sur « Molecular tools ». Cliquez ensuite sur le bouton « delete » situé à côté de l'outil moléculaire à supprimer.

Une fenêtre demandant votre confirmation apparaîtra. Attention, lorsque vous confirmez la suppression, l'action sera irréversible et vous ne pourrez plus accéder aux données concernant cet outil moléculaire.

3.1.4. Éditer un outil moléculaire

Si vous avez besoin d'éditer un outil moléculaire déjà présent dans la base, il vous faudra aller dans le menu situé à gauche de la page dans la section « search tools ». Cliquez ensuite sur « Molecular tools ». Trouvez l'outil moléculaire qui vous intéresse dans la page puis cliquez sur le bouton « edit » situé à côté de lui.

La page d'édition de l'outil moléculaire s'affichera (figure 45).

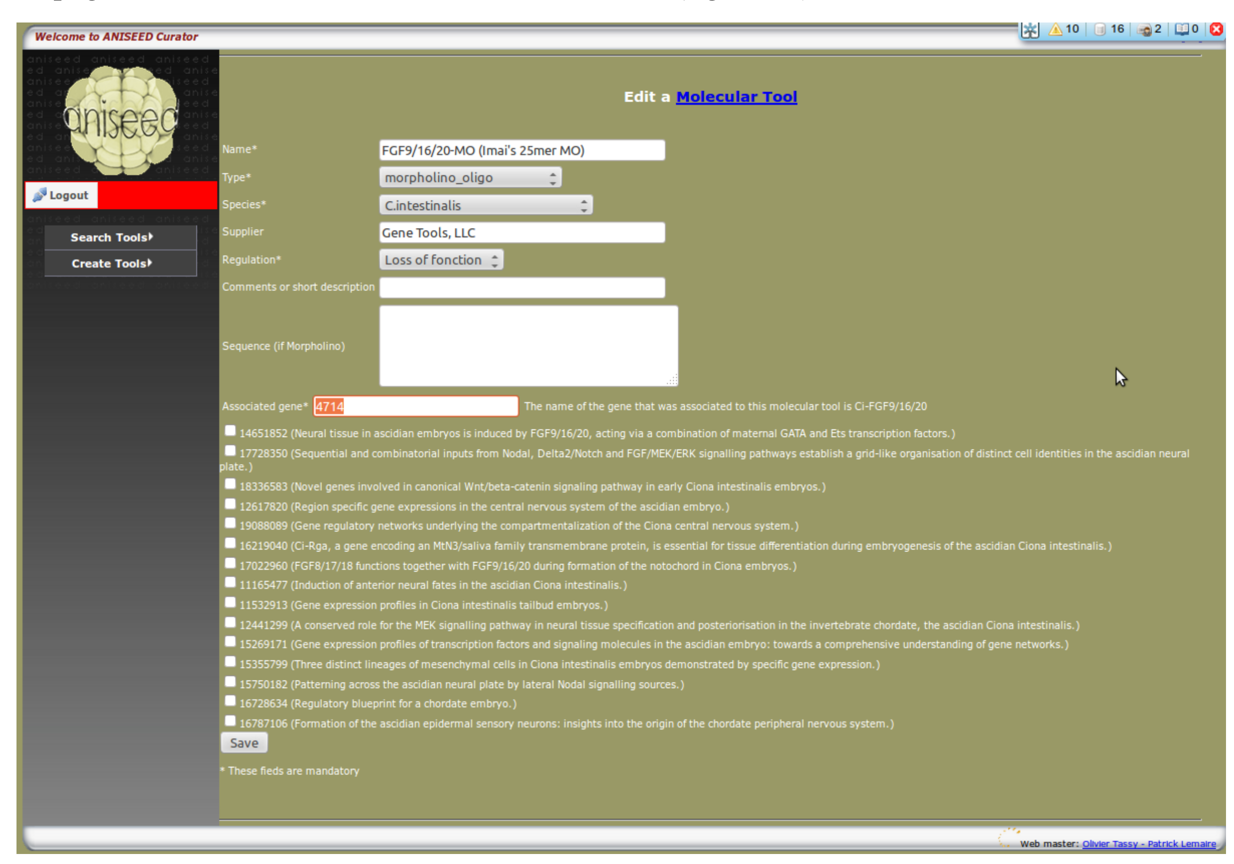


Figure 45 - Modifier un outil moléculaire

Cette page contient tous les champs pré remplis relatifs à l'outil moléculaire que vous avez sélectionné. Vous pourrez modifier ou remplir des champs vides. Vous pourrez également cocher des publications ou en décocher s'il y a eu erreur.

Enfin, si vous voulez changer de gène, il vous suffira de modifier la donnée entrée dans le champ « associated gene ». Les publications associées à ce gène apparaîtront ensuite.

3.2. Les in situ

3.2.1. Créer l'in situ

Imaginons que vous souhaitez insérer dans la base une expérience ISH montrant un embryons au stade 110 cellules, dérivé d'un œuf injecté de morpholino FGF9/16/20 et sondé par une expression Ci-Bra. Vous devrez effectuer plusieurs étapes afin d'enregistrer cette expérience.

Dans un premier temps, dirigez vous vers le menu situé à gauche de la page et allez dans la section « create tools ». Choisissez ensuite l'onglet « Create an in situ data ».

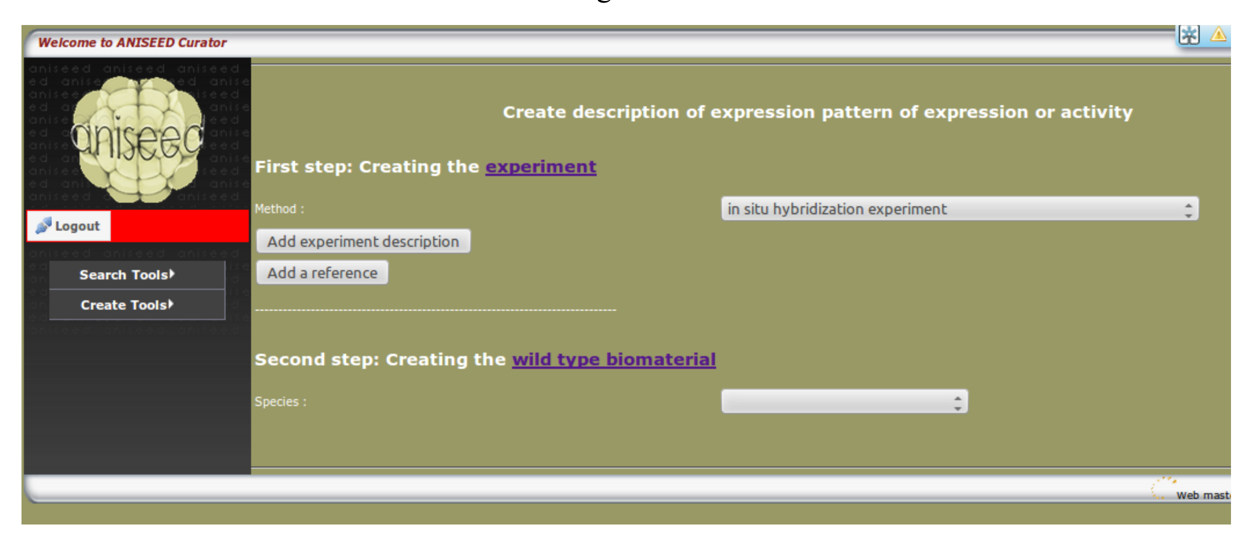


Figure 46 - Créer l'in situ (expérience et biomatériel de type sauvage)

La page qui apparaîtra (figure 46) vous permettra de créer « l'expérience » qui est en fait un objet informatique comprenant la méthode d'expérimentation, la description d'une expérience, les publications associées à cette expérience. Elle vous permettra également de définir le biomatériel qui regroupe toutes les informations concernant l'échantillon sur lequel vous allez effectuer votre expérience. Le biomatériel comprend l'espèce et le stade de développement de l'échantillon.

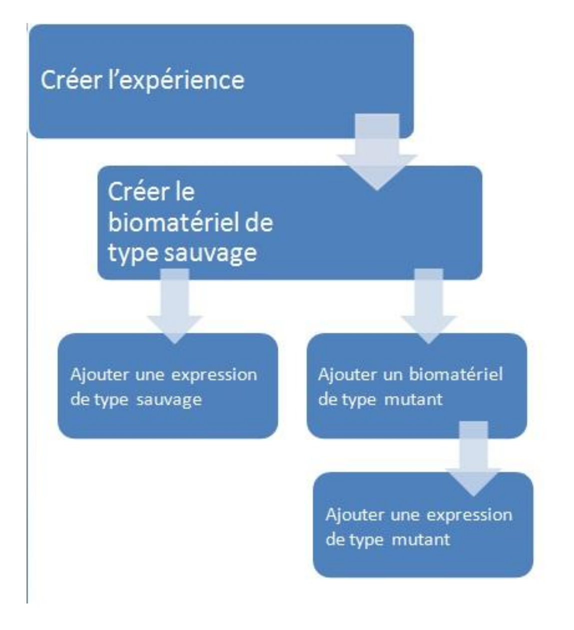


Figure 47 - Créer un in situ

Voici une brève présentation de l'architecture globale des pages (cf. figure 47) : Vous commencez par insérer les données liées à l'expérience et le biomatériel de type sauvage. Vous pourrez ensuite ajouter des profils d'expression géniques liées à ce biomatériel de type sauvage. Vous pourrez aussi ajouter un biomatériel de type mutant (et donc spécifier les molécules dérégulées par celui-ci, ainsi que les modifications embryologiques effectuées). Après avoir créé ce biomatériel, vous pourrez y associer des profils d'expression.

Commençons par créer l'expérience :

- Choisissez la méthode d'analyse de votre expérience dans la liste déroulante.
- Ensuite, vous pouvez éventuellement ajouter une description en cliquant sur « add experiment description ». Un champ texte apparaîtra que vous pourrez ensuite cacher (si vous le trouvez trop encombrant) en cliquant sur « Hide ».
- Vous pouvez ensuite associer des publications à votre expérience en cliquant sur « add a reference ». Lorsque vous commencerez à saisir le nom de votre publication, vous pourrez être aidé par le mécanisme d'auto complétion. Si vous voulez associer d'autres publications à votre expérience, cliquez sur « add more references » et un champ supplémentaire apparaîtra. Vous pouvez saisir autant de publications que nécessaire en dupliquant le champ à chaque fois.

Maintenant créons le biomatériel de type sauvage :

- Choisissez une espèce dans la liste déroulante. Lorsque vous aurez choisi une espèce, un arbre contenant tous les stades de développement apparaîtra automatiquement.
- Vous pouvez maintenant choisir le stade de développement auquel l'analyse de l'expression des gènes a été effectuée. Pour faire déplier l'arbre complètement et donc voir toutes ses feuilles, cliquez sur « Expand all ». Pour faire l'action contraire et donc ne voir que les nœuds, cliquez sur « Collapse all ». Vous pouvez également n'ouvrir et ne fermer que certains nœuds en jouant avec le triangle placé à côté de chaque stade. Une fois le stade choisi, le bouton « save » apparaîtra.

La nouvelle page qui apparaîtra résume les informations liées à l'in situ qui sont contenues dans la base de données. (cf. figure 48). Sur cette page, de nombreux liens permettent d'accéder à d'autres pages permettant d'éditer l'in situ, d'ajouter un nouveau biomatériel de type mutant, d'ajouter une expression au biomatériel de type sauvage...

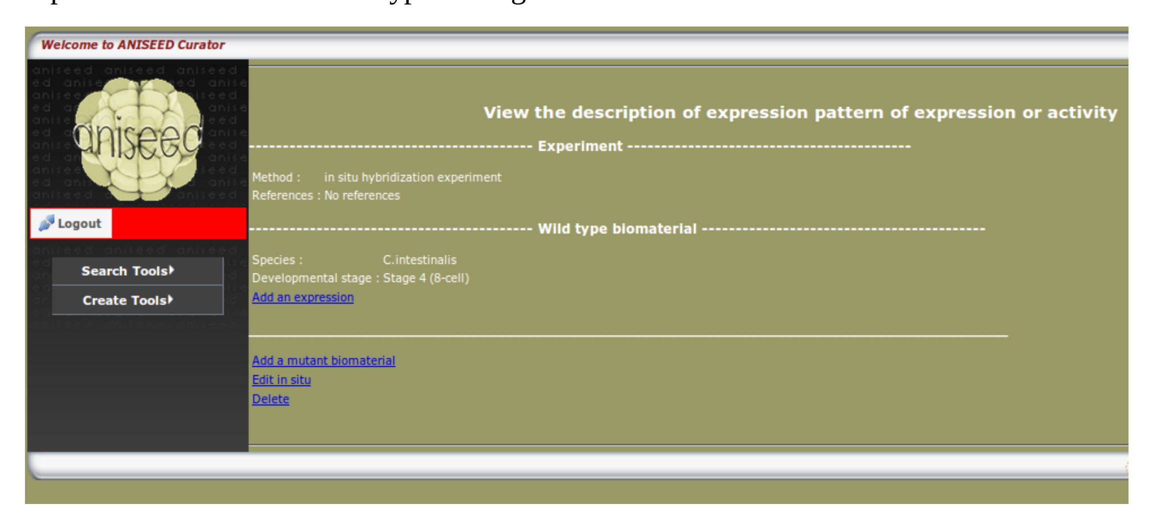


Figure 48 - Résumé de l'in situ

3.2.2. Ajouter un biomatériel de type sauvage

Afin de créer un biomatériel de type sauvage, il faudra renseigner la molécule dérégulée ou/et la manipulation d'embryon qui a été faite. En effet, l'espèce et le stade de développement sont présélectionnés puisqu'ils doivent être les mêmes que pour le biomatériel de type sauvage.

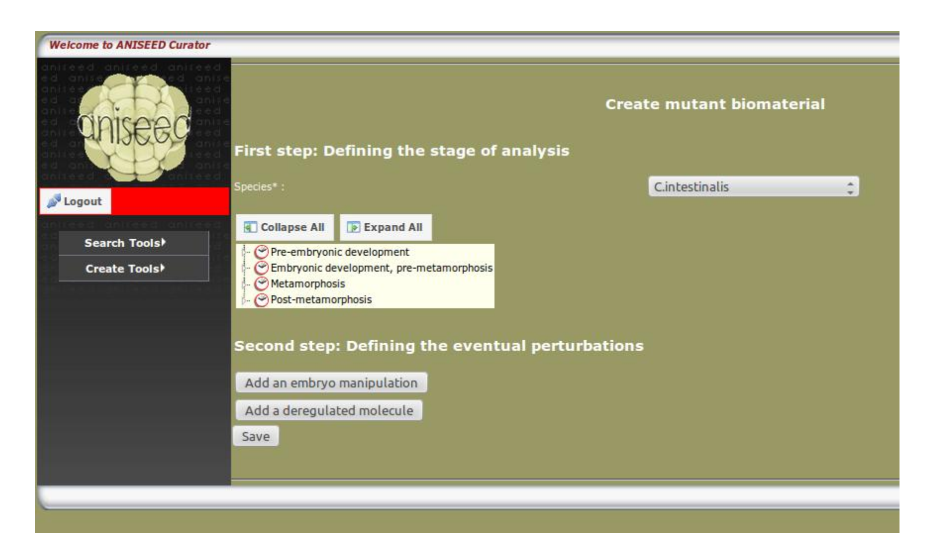


Figure 49 - Créer un biomatériel

- Pour associer une manipulation d'embryon à votre biomatériel, cliquez sur le bouton « add an embryo manipulation ». Des champs dans lesquels vous pourrez renseigner le type de manipulation et le stade de développement apparaîtront. Lorsque vous aurez sélectionné un stade de développement, une fenêtre apparaîtra, contenant un arbre des territoires anatomiques. Sélectionnez ensuite les territoires de l'embryon qui sont concernés par la manipulation.
- Pour associer une molécule dérégulée à votre biomatériel, cliquez sur le bouton « add a deregulated molecule ». Deux arbres contenant des stades de développement et un champ de type texte apparaîtront. Dans le premier arbre, vous pourrez sélectionner le stade de début de la dérégulation. Dans le second, sélectionnez son stade de fin. Dans le champ « molecular tool » renseignez le nom de l'outil moléculaire. L'auto complétion vous aidera à trouver le nom exact de l'outil moléculaire. Lorsque vous aurez saisi cet outil moléculaire, le nom du gène y étant lié apparaîtra également.

3.2.3. Ajouter une expression de type sauvage ou perturbé

Pour rajouter un profil d'expression à votre biomatériel, cliquez sur « add an expression » en dessous du biomatériel concerné.

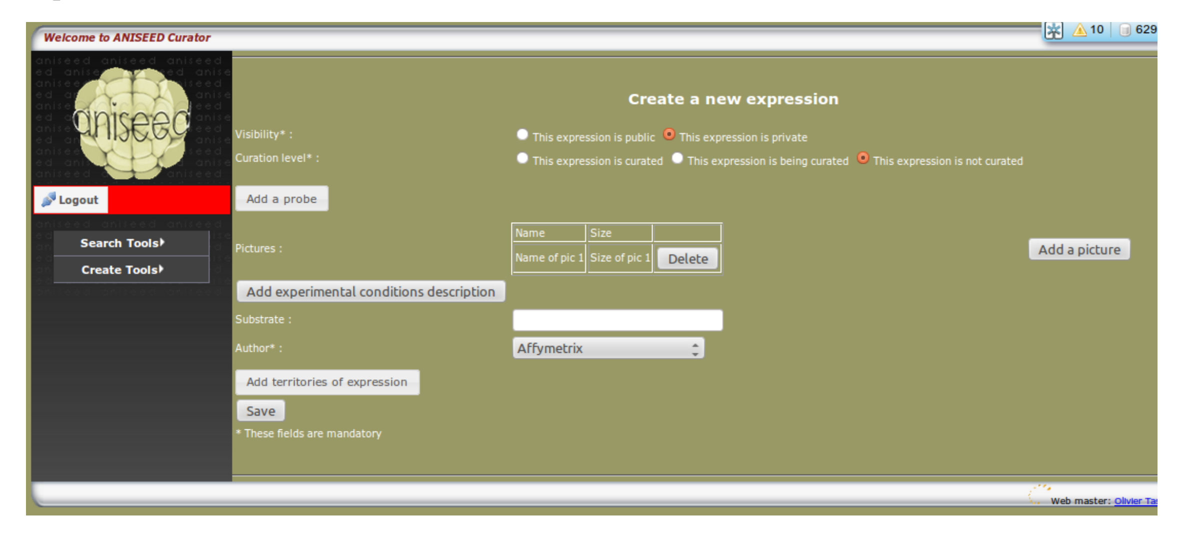


Figure 50 - Créer une expression

Vous accéderez ainsi à la page « Create a new expression ».

- Vous commencerez par définir le niveau de visibilité de votre expression : Si l'expression est destinée à rester entre vos mains et celles de votre labo, cochez « this expression is private ». Si ce n'est pas le cas et que vous souhaitez que tout le monde y ait accès, choisissez plutôt « this expression is public ».
- Définissez ensuite le niveau de curation de votre expression. Si l'expression n'est pas en état de curation, laissez le choix « this expression is not curated ».
- Pour définir un profil d'expression, il faut définir une sonde que vous voulez associer à votre expression en cliquant sur « add a probe ». Aniseed n'accepte que les sondes enregistrées dans Genbank. Une fenêtre s'affichera (figure 51). Commencez par saisir le gène dont le profil d'expression est étudié dans le champ prévu à cet effet. Lorsque vous aurez fait votre choix, une liste de clones apparaîtra automatiquement. Sélectionnez dans cette liste le clone à partir duquel la sonde a été réalisée.

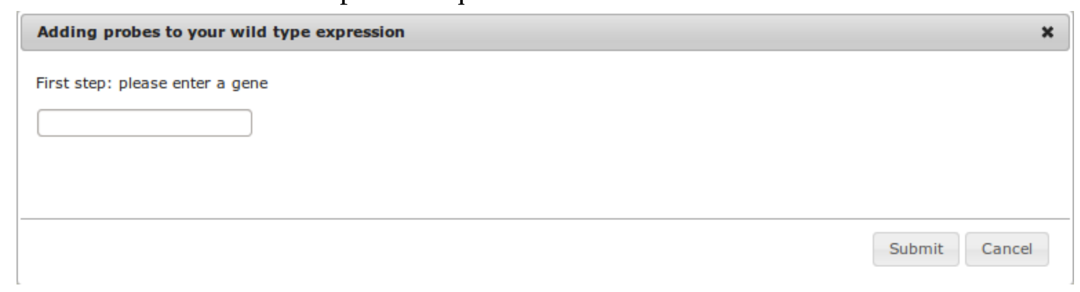


Figure 51 - Fenêtre pour sélectionner la sonde

- Vous pouvez ensuite ajouter des images liées à votre expression. Pour cela, cliquez sur le bouton « add a picture ». Le bouton « parcourir » vous permettra d'aller chercher votre image sur votre ordinateur. Si vous souhaitez associer une autre image à votre expression, cliquez sur « add more pictures ». Lorsque vous aurez cliqué sur ce bouton, un nouveau champ apparaîtra. Dans ce champ vous pourrez associer une autre image à votre expression.
- Si vous voulez ajouter une description des conditions expérimentales cliquez sur le bouton « add experimental conditions description ».
- Saisissez le substrat utilisé dans l'expérience.
- Sélectionnez un auteur pour cette expression. Si vous entrez des données issues d'un article, choisissez le premier auteur.
- Vous pouvez ensuite spécifier les territoires anatomiques marqués en cliquant sur « add territories of expression ». Une nouvelle fenêtre s'affichera (cf. figure 52). Cette fenêtre comprend un arbre avec tous les territoires anatomiques liés au stade de développement choisi. À côté de chaque territoire que vous sélectionnerez, vous pourrez préciser s'il est « not sure », ce qui signifie que vous n'êtes pas sûr qu'il y ait bien ce territoire dans les territoires d'expression. Vous pourrez également spécifier si le territoire est « part of » ce qui veut dire que la tâche est réservée à une partie du territoire. Vous pouvez également préciser la position subcellulaire des marquages d'un territoire.

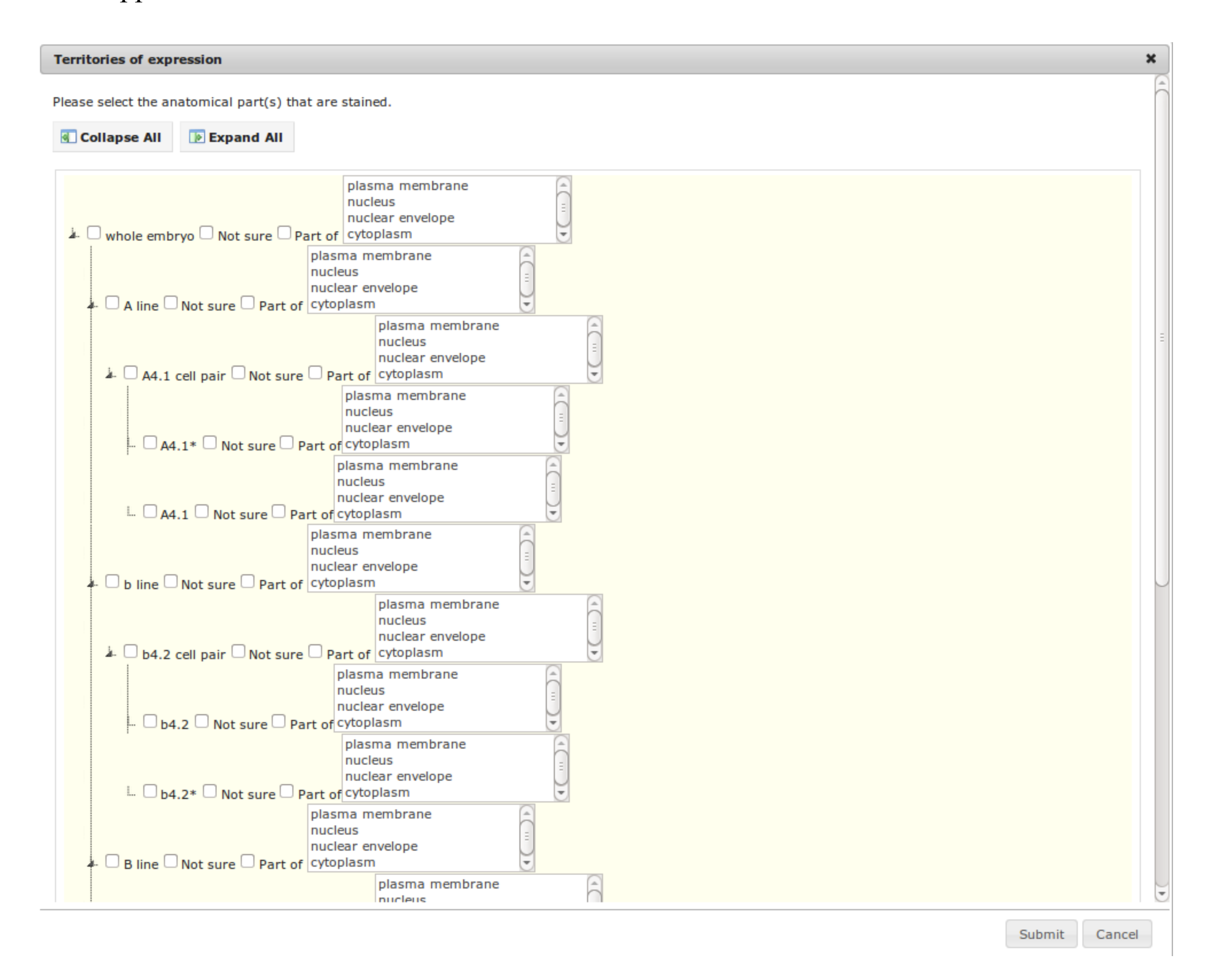


Figure 52 - Fenêtre pour ajouter des territoires d'expression

✦ Lorsque vous aurez renseigné toutes les données, vous pourrez enregistrer votre profil d'expression.

4. Rapport d'activité

4.1. Cycle de développement

Pour le développement de l'interface de curation, j'ai décidé de suivre un cycle itératif (cf. figure 53 visible ci-dessous).

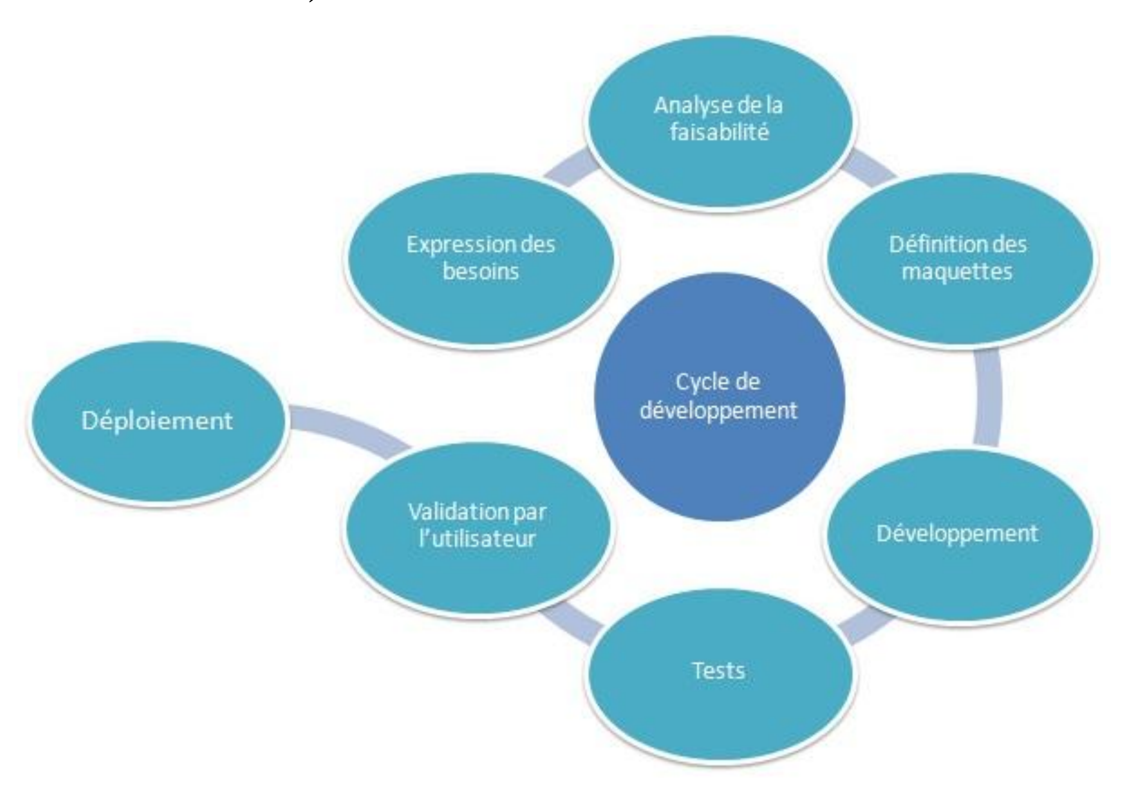


Figure 53 - Cycle de développement

Le développement s'est divisé en plusieurs étapes :

- La première étape consistait à définir les besoins des utilisateurs. Cette étape se fait en étroite collaboration avec l'utilisateur pour bien le comprendre, et inclut la compréhension de notions biologiques.
- La deuxième étape consistait à déterminer la faisabilité de ce que l'utilisateur souhaitait. Cette étape a été l'une des plus délicates car je savais que j'aurais besoin d'utiliser des notions d'informatique nouvelles. (Par exemple, je savais qu'il me faudrait utiliser Ajax pour répondre à certains besoins, mais n'ayant à l'époque aucune connaissance en Ajax, je ne savais pas quelles étaient ses limites.)
- Par la suite, j'ai essayé de produire des maquettes pour bien valider ce qu'il faudrait produire.
- Ensuite j'ai développé la page correspondant à la maquette. Cette étape était la plus longue car elle comprenait aussi l'étape suivante qui regroupait les tests. En effet, j'ai décidé de faire des tests au fur et à mesure du développement.
- Après que la page est entièrement fonctionnelle je la faisais valider par la curatrice et Patrick Lemaire. Parfois il venait s'ajouter des détails en plus, oubliés ou non spécifiés au départ. Cette étape faisait donc parfois revenir à l'étape de départ pour rajouter de nouvelles fonctionnalités aux pages.

4.2. Planification

Lors de la première semaine de stage, j'ai principalement découvert Aniseed à travers des articles, et de la documentation. J'ai pu voir le fonctionnement global du modèle Chado, la base de données d'Aniseed, ainsi que le Framework que j'allais utiliser. Pour mieux comprendre Jelix j'ai fait quelques tutoriels présents sur le site. J'ai développé un module de news pour m'entrainer (Création, édition et suppression). J'ai ainsi pu découvrir comment s'articulaient les templates, les contrôleurs et les classes sous Jelix.

Sur la copie en local de la version 4 d'Aniseed, j'ai pu ensuite m'entrainer à essayer de modifier les couleurs, et les requêtes des différentes pages. J'ai ainsi pu voir plus en détail où étaient situés les différents blocs de code et l'utilité des modules. Grâce à cet exercice, j'ai également pu manipuler un peu la base de données.

À la fin de cette première semaine, Cyril Martin et Patrick Lemaire m'ont présenté le vrai site d'Aniseed visible sur internet ainsi que l'interface de curation. Ils m'ont expliqué en détail le sujet du stage et quelles allaient être mes tâches. Cyril m'a également donné quelques pistes sur ce qu'il faudrait que j'utilise pour parvenir à ces objectifs.

La première tâche qui m'a été confiée a été de faire la page permettant de créer un nouvel outil moléculaire. J'ai entrepris de faire une première version simplifiée de la page, puis petit à petit j'ai rajouté les différentes fonctionnalités qui m'avaient été demandées. Au total pour cette page, j'ai développé six versions. J'ai choisi de développer par versions, et non de faire progresser une même page car selon les versions je testais des technologies différentes (HTML5, Jquery, Ajax…).

J'ai régulièrement discuté avec Delphine Dauga (la biocuratrice d'Aniseed) par mail pour définir exactement les besoins de curateur. En effet, il ne m'a pas été demandé de faire une page précise, on m'a juste défini les besoins généraux des utilisateurs. Je devais donc me renseigner par exemple sur quels champs étaient obligatoires, combien de gènes on pouvait choisir pour un outil moléculaire… Ces informations paraissent souvent peu importantes pour les non-informaticiens, alors qu'elles sont cruciales pour savoir les requêtes, quels éléments d'option sont utilisables dans les formulaires…

Il m'a fallu un mois pour aboutir à la dernière version fonctionnelle. Cette page m'a apporté énormément de connaissances nouvelles : Déjà, c'était la première fois que je codais en suivant un Framework (avant je ne suivais que le design pattern MVC), j'ai appris à naviguer et manipuler une énorme base de données (plus de 200 tables), j'ai appris à utiliser jQuery, m'initier un peu à JavaScript, j'ai aussi appris à utiliser des outils comme FireBug pour débugger, je me suis familiarisée avec l'IDE NetBeans que je ne connaissais pas (avant je codais sur gedit ou notepad++). J'ai aussi pu utiliser SVN pour me coordonner avec mon tuteur.

J'ai dû également faire preuve d'autonomie. En effet, mon tuteur a du s'absenter durant ma deuxième, troisième et quatrième semaine de stage. J'ai posté régulièrement sur Redmine des rapports relatant l'avancée de mon travail, afin qu'un autre développeur soit en mesure de comprendre mon code.

En dehors de ces compétences en informatique, j'ai également pu voir comment se déroulait le travail au CNRS. En effet, je n'avais jamais assisté auparavant à des réunions de labo. J'ai aussi pu voir l'importance des informaticiens dans le milieu de la biologie.

Une des choses les plus fastidieuses lors de mon stage a été la recherche d'informations. J'ai passé de longues heures à chercher sur internet pour me renseigner sur quels outils existaient pour mettre en place les fonctionnalités qu'on m'avait demandées. Grâce à ces recherches, j'ai découvert et appris à utiliser de nombreux outils et concepts, ainsi que des rudiments dans divers langages.

Lorsque j'ai fini ma première page, j'ai dû présenter mon travail devant toute l'équipe Lemaire lors d'un lab meeting. Comme ces réunions s'adressent à des biologistes et des informaticiens, il faut essayer de s'exprimer clairement afin que notre travail soit compréhensible et intéressant pour tous.

Ensuite, suite à une deuxième petite réunion avec Patrick Lemaire et Cyril Martin, nous avons défini ma seconde mission. Pour cette page, je devais recréer une page du curateur qui posait des problèmes dans la version précédente. En effet, ce n'était pas une amélioration de l'ergonomie qui devait être faite mais plutôt une réorganisation des pages. Pour enregistrer un in situ complet, il faut insérer beaucoup de données dans la base qui sont liés à plusieurs tables différentes. De plus, il a fallu intégrer la notion d'expérience qui n'existait pas dans la v3.

Ce qui a été difficile dans ma deuxième mission a surtout été de comprendre le besoin des utilisateurs. En effet, j'ai parfois reçu quelques informations contradictoires selon les sources. J'ai donc souvent développé des fonctionnalités qui finalement n'ont pas eu d'utilité. J'ai trouvé très intéressant de voir les divergences des opinions, et la difficulté de comprendre le besoin exact des utilisateurs.

Ces pages étaient techniquement plus difficiles. Maintenant que je m'étais familiarisée avec le fonctionnement de Jelix et de la base de données, je devais apprendre à utiliser de nouvelles fonctionnalités de Jquery. De plus, il me fallait absolument comprendre quelques notions de biologie plus complexes pour comprendre les différents objets et les liens qui existaient entre eux. Je n'ai pas fait plusieurs versions de cette page car il n'y avait pas de choix à faire quant aux technologies à utiliser. Lorsqu'elle était presque finie, j'ai également présenté cette page en lab meeting.

4.3. Méthodes et outils de travail

Lors de mon stage j'ai découvert une nouvelle méthode de travail alliant autonomie et travail d'équipe. Les problèmes ou difficultés techniques que j'ai pu rencontrer ont généralement été résolus en recherchant sur internet. Toutefois, le chef de l'équipe, Patrick Lemaire, passait régulièrement dans le bureau informatique pour voir l'avancement des projets, donner son avis, des conseils, ou encore compléter les consignes données à l'origine. Je devais également rapporter sur Redmine (une application web libre de gestion de projets) l'avancement de mon travail, l'explication du code et un résumé des solutions apportées aux différents problèmes rencontrés. Cette documentation écrite en anglais permettra aux autres développeurs de comprendre mon travail, de ne pas être confronté aux mêmes erreurs, et de

pouvoir maintenir l'interface plus facilement. Sur cet outil, j'essayais aussi de planifier les dates de fin de chaque tâche afin d'améliorer mon organisation. Le diagramme de Gantt suivi pendant mon stage est visible en annexe 7.

J'ai eu la chance de discuter plusieurs fois avec Delphine Dauga, la curatrice d'Aniseed. Ces conversations m'ont permis de trouver des idées sur comment simplifier son travail et donc de rendre l'interface de curation la plus agréable possible. Nous avons ainsi pu définir ensemble ses besoins. Lorsque j'avais une question d'ordre biologique, je pouvais facilement la contacter par mail.

Toutes les semaines, je participais aux réunions d'équipe où un membre de l'équipe présente l'avancement de son travail en anglais. Ces réunions permettent de voir sur quoi les autres personnes travaillent, de proposer des idées, et permettent à la personne qui présente de prendre du recul sur son travail. En effet, le présentateur doit essayer d'être le plus compréhensible possible. Sachant que l'équipe se compose de biologistes et d'informaticiens, il doit savoir se faire comprendre par des personnes n'ayant presque aucune formation dans le milieu où il travaille. Au cours de mon stage, j'ai eu à animer deux de ces réunions.

Conclusion

L'objectif principal de mon stage était de développer une partie de l'interface permettant la curation pour les biologistes. Cette interface s'intègre dans la version 4 d'Aniseed. À travers plusieurs formulaires les plus intuitifs possibles, l'utilisateur doit pouvoir saisir dans la base différents types de données biologiques, ainsi que pouvoir créer des liens avec des entités déjà existantes. Les objectifs qui m'ont été donnés en début de stage ont bien été atteints : une partie de l'interface de curation d'Aniseed version 4 est bien fonctionnelle, et pourra facilement évoluer grâce à l'utilisation d'une base de données bien construite, et au Framework Jelix. L'utilisation de Redmine permet d'avoir un suivi détaillé du stage et des explications sur le code. Les commentaires dans le code permettent la réutilisation et la maintenance de mon travail, et facilitent le développement des autres pages de l'interface de curation.

Au début du stage, j'ai dû apprendre et intégrer rapidement des notions en biologie qui m'étaient nécessaires pour comprendre l'interface que je devais créer. De plus, n'ayant jamais travaillé avec une si grande base de données et un framework auparavant, l'adaptation était assez éprouvant. J'ai aussi été confrontée à des problèmes de communication au sein de l'équipe. À certains moments, selon le biologiste, l'information était différente. J'ai également pu constater à quel point il peut être difficile de communiquer avec des personnes extérieures au monde de l'informatique. Durant ce stage, j'ai rencontré plusieurs difficultés qui m'ont cependant permis d'acquérir des compétences et des connaissances supplémentaires.

Premièrement, ce stage m'a permis d'améliorer mes compétences en programmation web. J'ai appris à utiliser des outils que je ne connaissais pas comme NetBeans pour faciliter le codage, Firebug pour aider à debugger, Redmine pour mieux organiser l'avancement du projet... J'ai aussi appris à utiliser le Framework Jelix, le système de gestion de base de données postgreSQL, et enfin SVN. Sur le plan humain, je me suis intégrée très rapidement à cette équipe qui sait travailler de manière productive tout en restant décontractée. Durant ce stage, j'ai pu développer mon autonomie et apprendre à mieux rechercher les réponses à mes questions dans de la documentation. Je sais également mieux expliquer mon travail grâce aux réunions d'équipe et aux résumés à poster sur redmine. Finalement, j'ai pu découvrir beaucoup de choses (des outils, des langages…) mais également mettre en pratique les connaissances théoriques que j'ai acquises durant ma formation. De plus, je me suis confrontée aux difficultés réelles du monde du travail.

Le bilan de ce stage est donc très positif.

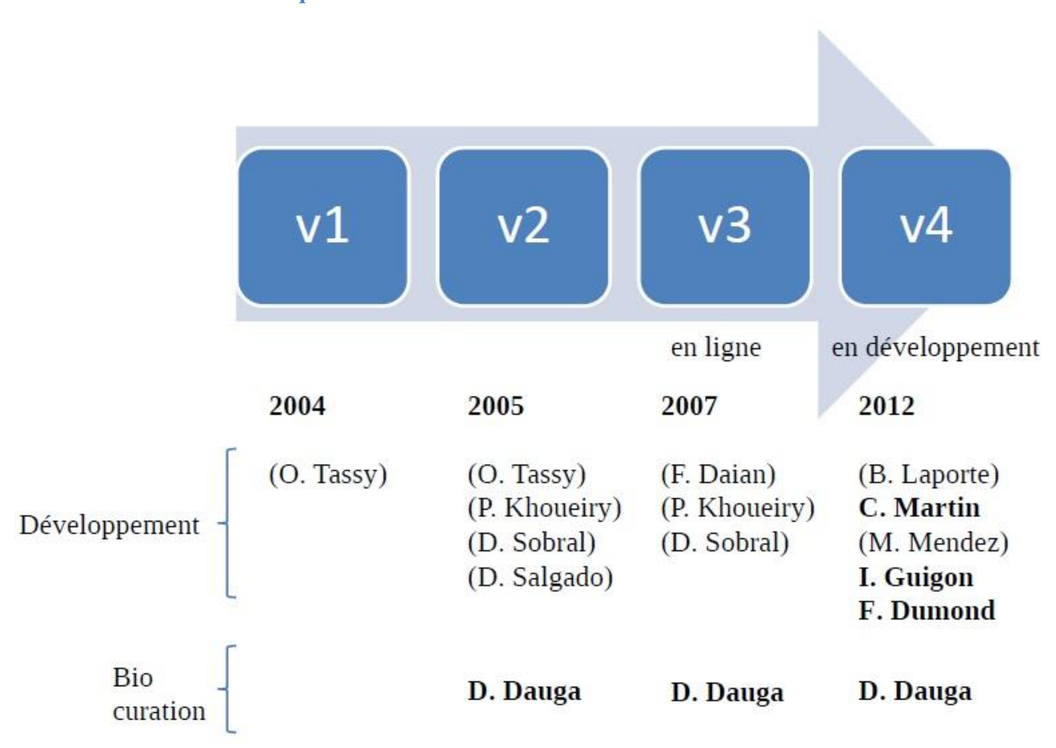
Sitographie

SITES INTERNET CONSULTÉS

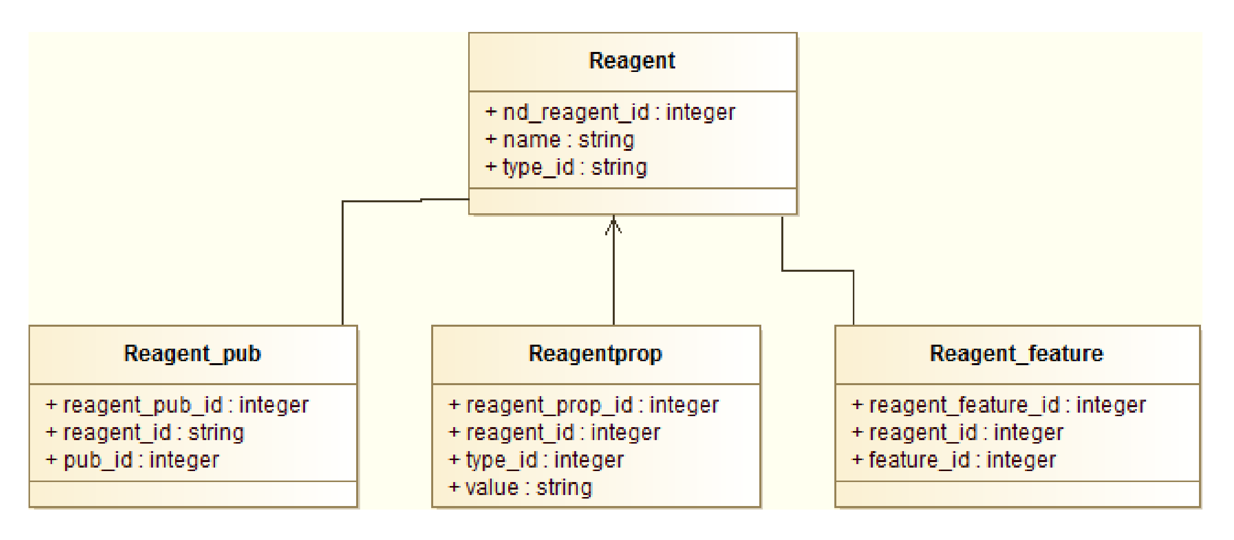
Adresse de la page	**Type de site**	**Information recherchée**
http://precedings.nature.com/documents/3186/version/1	Site de publications de chercheurs	Informations sur aniseed
http://bioinformatics.oxfordjournals.org	Site de publications	Informations sur Chado
http://docs.jelix.org/fr/manuel-1.4	Manuel de Jelix 1.4.	Documentation sur Jelix
http://docs.jelix.org/fr/manuel-1.5	Manuel de Jelix 1.5.	Documentation sur la version de Jelix utilisée par Aniseed après la mise à jour
http://jelix.org/reference/1.5.2/	API	API de Jelix version 1.5.2
http://jqueryui.com	Site officiel de Jquery UI	Documentation sur Jquery ui, demos des widgets, des méthodes…
http://siteduzero.com	Site communautaire	Informations sur Ajax
http://api.jquery.com	Site officiel de Jquery	Documentation sur Jquery, API
http://xul.fr/xml-ajax.html	Site de tutoriels	Tutoriels et demos Ajax
http://www.grafikart.fr/tutoriels/jquery	Site de tutoriels	Tutoriel sur Jquery
http://ajax.developpez.com/cours/	Cours sur Ajax	Cours sur Ajax
http://forum.phpfrance.com/faq-tutoriels	Forum	Informations sur les listes liées
http://siddh.developpez.com/articles/ajax/	Site de tutoriels	Informations sur Ajax
http://codular.com/jquery-autocomplete-suggestion-dropdown	Site proposant des articles sur différentes notions d'informatique	Article sur l'auto complétion en utilisant Jquery
http://www.javascriptfr.com/	Site de tutoriels	Informations sur les listes liées, ajax, et jquery
http://www.kommunauty.fr/	Site communautaire	Tutoriel sur l'auto complétion (javascript)
http://www.dynamicajax.com/fr/index.php	Site sur ajax	Informations sur Ajax
http://www.blog-nouvelles-technologies.fr/	Site proposant des articles sur diverses notions d'informatique	Explications sur le concept global d'auto complétion
http://desgeeksetdeslettres.com	Site d'informatique	Idées pour améliorer son auto complétion
http://plusdescripts.fr/	Site d'informatique	Tutoriel sur le clonage de champs grâce à Jquery
http://phpprogramming.wordpress.com/	Wordpress	Tutoriel sur le passage de valeurs cachées
http://www.tomsyweb.com	Blog	Tutoriel sur le masquage de champs (javascript)
http://gmod.org/wiki	Site de gmod	Documentation sur Chado
http://christophe.kerhousse.free.fr	Site d'informatique	Explications sur Ajax
https://developer.mozilla.org/fr/docs/JavaScript	Documentation	Documentation de JavaScript

Annexes

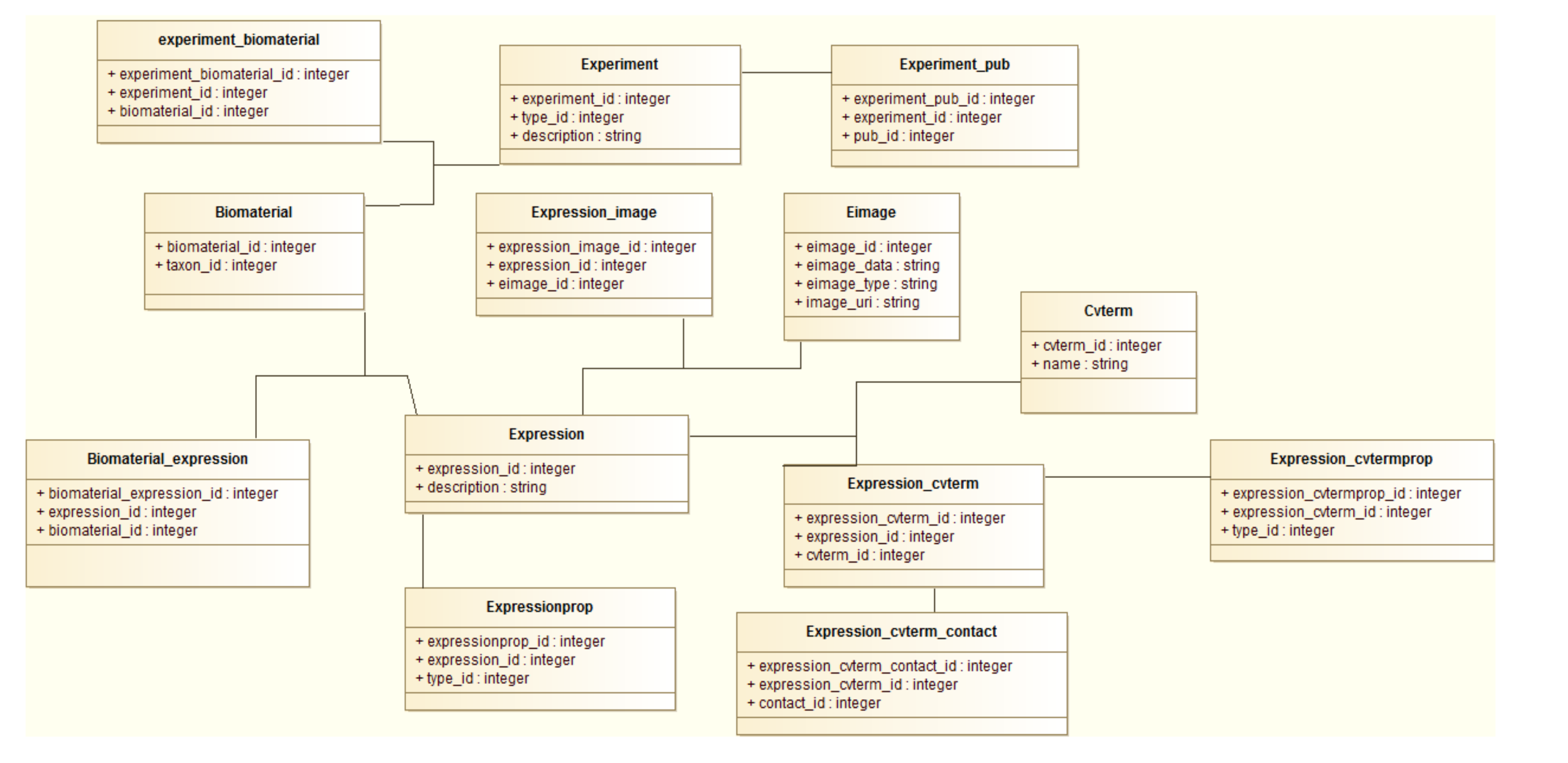
Annexe 1 - Historique d'Aniseed



Annexe 2 - Diagramme de classe Outils moléculaires

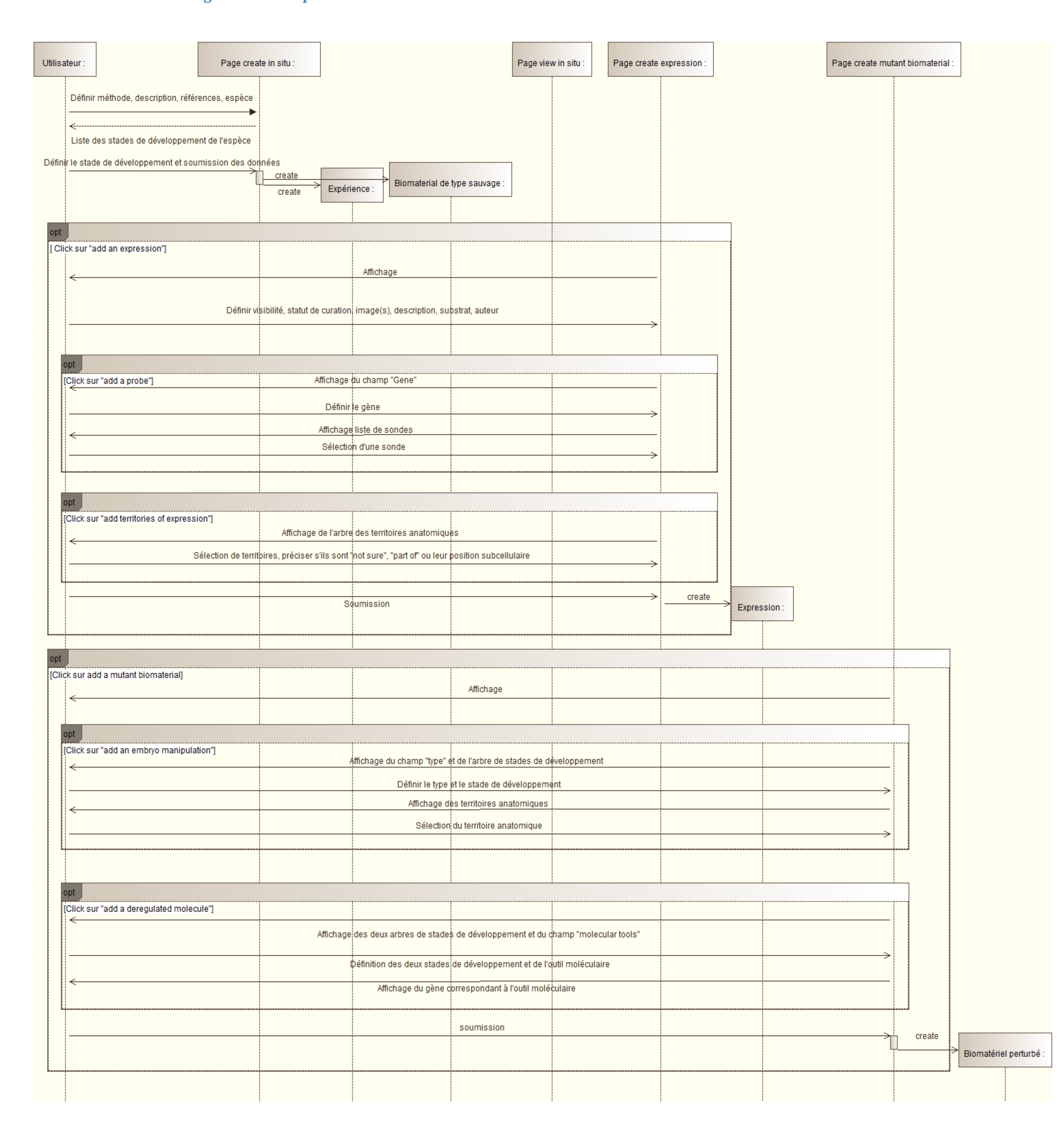


Annexe 3- Diagramme de classe In situ

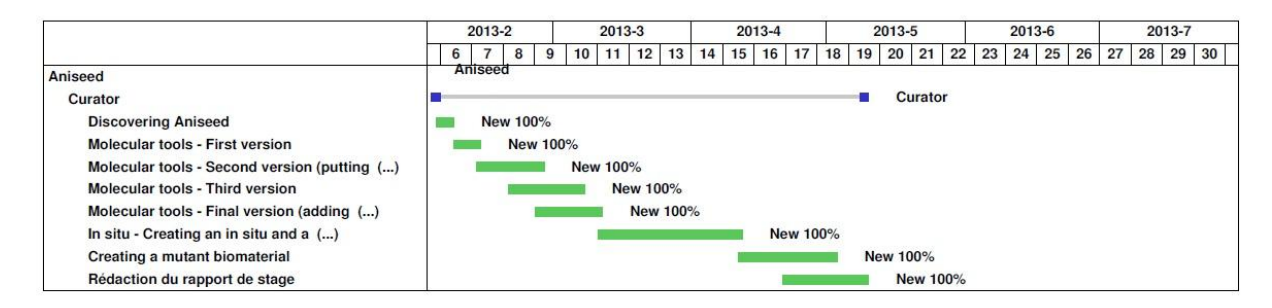


Experiment: Sert à définir l'expérience.
Experiment_pub: Sert à créer des liens entre une expérience et des publications.
Experiment_biomaterial: Sert à lier l'expérience à son biomatériel.
Biomaterial: Sert à définir un biomatériel
Biomaterial_expression: Sert à lier le biomatériel à ses expériences.
Expresssion: Sert à définir une expression
Expressionprop: Sert à enregistrer des informations liés à une expression (le substrat, les commentaires...)
Expression_image: Sert à faire le lien entre une expression et ses images
Eimage: Sert à définir une image
Cvterm : Sert à définir un cvterm (vocabulaire contrôlé)
Expression_cvterm: Sert à créer des liens entre des cvterms et une expression.
Expression_cvtermprop et Expression_cvterm_contact: Servent à ajouter des spécifications au lien expression_cvterm

Annexe 4 - Diagramme de séquences de création d'un in situ



Curator



Annexe 5 - Diagramme de Gantt

Résumé

Mon stage s'est centré sur le développement d'une section de la version 4 du site d'Aniseed. Cette section concerne le curateur qui est l'interface permettant aux biologistes d'insérer leurs données dans la base de données d'Aniseed. Cette interface s'articule autour d'une série de formulaires conviviaux et intelligents, développés grâce à des plugins comme Jquery ou Jstree.

Ce site a été développé en suivant le Framework Jelix. Sa base de données s'appuie sur le modèle Chado. Le site a été créé en utilisant plusieurs langages dont HTML, PHP, JavaScript…

Mots clés

Développement web, Curation, Aniseed, Jquery, Jelix, Chado

Summary

My internship was centered on the development of a part of Aniseed's fourth version. This part concerns the curator which is the interface that allows biologists to enter their data in Aniseed's database. This interface is formed of a succession of user-friendly forms, developed thanks to plugins like Jquery or Jstree.

This website was developed following the Jelix framework. The database uses the Chado model. The site was created using several languages such as HTML, PHP, JavaScript…

Key words

Website development, Curation, Aniseed, Jquery, Jelix, Chado